Representing Knowledge Patterns in A Conceptual Database Design Aid: A Dual-Base Knowledge Model

by

CHANG, Tsair-Yuan
Diploma, MBA

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

University of Warwick
Warwick Business School

June, 1998

Contents

Chapter 6

Example and Discussion 163

Appendices

Tables

Figures

Acknowledgements

My thanks are due to those people who have encouraged my study, enlightened my knowledge, and shared my feelings during my knowledge-exploring life.

First, I would like to thank Rober Hurrion for giving me guidance and encouragement when I needed it, and for reading countless drafts of the thesis. His attitude to my research has always been kind to me and he gave me the courage to embody my ideas step by step. My deep appreciation also goes to Dr. Godwin, a committee member of my doctoral thesis viva, whose suggestions significantly inspired me to reshape the formalisation of the dual-base knowledge structure in my study. I am also indebted to Yasmin Merali for the time she spent advising me in the study of the rule-based school. Her suggestions made this research more complete through a discussion of the current progress of data modelling work. Special thanks go to Mr. Keith Taylor, who was very thoughtful and patient in proof-reading my thesis.

For enlightening me in the rhythm of *Tao* (☯), my deep thanks also go to Mr. Wu, Hui-Fu and Mr. Fu, Tsung-Li, who taught me Taoist philosophy, which originated in ancient China. They spent a lot of time helping me to realise how the *yin* (the dark part and the bright dot) and the *yang* (the bright part and the dark dot) interplay dynamically to fulfil the *Tao* function, and this insight provided me with the idea of two synergistic bases in which the concept base (*yang base*) and the case base (*yin base*) complement each other and work together. Also, thanks to my all friends for their help in various aspects of my work as well as their friendship in sharing my sorrow and joy: in particular Chien-Ho Wu, Meng-Yen Lin, Tzu-Chuan Chou, Chien-Wei Chen, Jau-Rong Chen, Chien-Ming Ting, Wei-Hwa Chen, Ming-Chen Shai and Hsin Lu.

My greatest debt, of course, belongs to those wise and progressive scholars who have led the explorations in knowledge discussed in this thesis. Many of them are named herein, but others are not. To all of them go both my thanks and my admiration.

Finally, my deepest gratitude goes to my mother, who has always instilled love in me, and my wife, Meng-Na, who has cared for our daughters, Jing-Fang and Yuh-Ing. Without their assistance and support, I could not possibly have passed through these most challenging years of my life.

CHANG, Tsair-Yuan

June, 1998

Declaration

This is to declare that:

1. I am responsible for the work submitted in this thesis.
2. This work has been written by me.
3. All verbatim extracts have been distinguished and the sources specifically acknowledged.
4. This work has not previously been submitted within a degree programme at this or any other institution.

Signature: CHANG, Tsair-Yuan

Date: 19 June 1998

Abstract

The current status of the Knowledge-Based Database Design Systems (KBDDSs) is reviewed. It is shown that they do not resolve the problems of the identification of the relevant objects (relations) and the interpretation of the identified objects from the semantic-rich reality.

Consequently, a theoretical architecture is developed to alleviate these problems by reusing the finished conceptual data schemata. By taking account of the essence of the reality and the problem-solving behaviour of experts, a new knowledge model called the Dual-Base Knowledge Model (DBKM), which involves two syngeristic knowledge structures, the concept and case bases, is constructed by the theories of conceptual knowledge in the psychological realm and the notions of relation and function from set theory. The aim is to provide rational and valid grounds for the support and interplay of these two bases in order to reuse the relevant old cases and facilitate the acquisition of new cases. Thus, the process model, which involves two process mechanisms, the case retrieval and knowledge accumulation mechanisms, is analysed according to the theory of the proposed DBKM. In this way, the feasibility of reusing the relevant schemata or part of them can be established in the DBKM architecture.

The functionality of the DBKM architecture is tested by a simulated example to show how the relevant cases are recalled in the knowledge pool and the new knowledge is stored in the knowledge repository. The distinctions between the DBKM architecture and the frameworks of current KBDDSs and Case-Based Reasoning (CBR) systems (from the knowledge-based system view), and between the DBKM and those knowledge models in current KBDDSs and rule-based data modelling approaches (from the knowledge-modelling view) are investigated to contrast the current levels of progress of the conceptual data modelling.

This research establishes the feasibility of the DBKM architecture, although it demonstrates the need to accommodate the dynamic and functional aspects of the Universe of Discourse (UoD). The main contributions of the DBKM are (1) to provide a valid basis for complementing the environments supported by the current KBDDSs and a rational basis for creating the symbiosis of humans and computer; and (2) to moderate the beliefs underlying the fact-based school and provide a hermeneutic environment, so that the confusion of the current conceptualising work can be alleviated and the difficulty of the conceptualising task can be eased to some degree.

[Keywords: Conceptual Database Design, Knowledge-Based Database Design Systems (KBDDSs), Dual-Base Knowledge Model (DBKM), Rule-Based Reasoning (RBR) Paradigm, Case-Based Reasoning (CBR) Paradigm, Conceptual Data Schema, Mental Model]

Chapter1

Introduction

1.1 Background

Conceptual database design is concerned with the conceptualisation of reality in terms of entities and relationships. There are two important contingent activities involved in this reality-conceptualising work: (1) identifying the relevant objects (relations) and interpreting the semantics of the identified objects from the semantic-rich reality; for instance, it is necessary to consider what kinds of objects are permitted in the *Universe of Discourse* (UoD), what roles they play and what contents they possess; (2) modelling the specifications according to the syntactics of a conceptual data model such as the entity, attribute and relationship primitives in the Entity-Relationship (ER) model [Chen, 1976] to specify the contents and roles of the desired objects in the specific UoD, which is the part of reality that is used to help specify the domain of interpretation.

Regarding data as a resource of the modern organisation, data analysis and modelling came to the forefront of the concerns of Information-Systems (IS) development in the 1970s [Lewis, 1994]. The pioneering work of Codd [1970], regarding the normalisation of relations, and Chen [1976], proposing ER

modelling, provide better ways to model the intrinsic structure of data, rather than using conventional files. Following the growing sophistication of data storage technologies, such as Data Base Management Systems (DBMSs), which allow a separation of the discussion of physical design issues, the process of database design was divided into three contingent and interactive phases: (1) conceptual design specifies the users' requirements and models the information needs in the form of a high-level DBMS-independent model (i.e. the conceptual schema); (2) logical design translates the conceptual schema into the data model of the target DBMS (i.e. the logical schema); (3) physical design transforms the logical schema into a form that is suitable for the specific hardware and DBMS (i.e. the physical schema) [Batini et al., 1992]. Thus, the conceptual design was recognised as the crucial phase for developing database systems, and then the question of how to help designers reflect the true structure of the organisation's data in the database became the main direction of enquiry.

Although several methodologies and software tools reported by Reiner [1992] have to date been proposed and developed to aid the conceptualising task, the primary focus of those proposals is on the syntactical aspect: how to provide more expressive syntactical tools for conceptual data modelling. That is, the database literature is heavily represented by theories (notations and methods) - the semantic data models [Peckham and Maryanski, 1988] - such as the ER model [Chen, 1976], Extended Entity-Relationship (EER) model [Teorey et al., 1986] and Semantic Database Models (SDM) [Hammer and McLeod, 1981; Hull and King, 1987], and provides no suggestion for overcoming the difficulties of object

identification and semantic interpretation, thus leading to the confusion of current conceptualisation work [Beynon-Davies, 1992; Lewis, 1994].

Traditionally, the conceptual design has always been carried out by database designers by interviewing the user-groups and analysing the existing documents and systems in order truly to understand and capture the essential nature and meaning of the data items and their structures in a specific UoD. Since the mid-1980s, Knowledge-Based Database Design Systems (KBDDSs), which use the concepts and ideas from the Artificial Intelligence (AI) discipline, have been developed to compensate for problems, such as the scarcity of human experts or lack of familiarity with the specific application domain, which are caused by the traditional approach [Storey and Goldstein, 1993]. Although the importance of object identification and semantic interpretation in this conceptualising task has been acknowledged, most efforts to develop these KBDDSs have still concentrated on the old approach - the syntactical aspect - and the more expressive and richer semantic integrity-constraining syntactical tools, for example by using the modus ponens (IF $\alpha \rightarrow \beta$ & α THEN β) to deduce the implicit facts. Leaving the tasks of object identification and semantic interpretation to the users (designers) results in the persistent problems of conceptualisation.

This study is concerned with investigating a knowledge model by taking account of the essence of the reality and the problem-solving behaviour of experts in order that the finished conceptual schemata or part of them can be reused and so help with the difficulties of the conceptualising task. This means that this research

seeks: (1) to set up an environment that provides the systematic correspondence between the ill-defined objects (relations) in the semantic-rich reality and well-defined entities (relationships) in the specified UoD to co-ordinate the multiplicity of the reality and to handle the complexity of the reality; and the holistic comprehension picture of the specified UoD, and the richer and more differentiated vocabulary of facets to assist the matching process of the visual, total pattern consisting of the bounded entities and relationships; and then (2) to test the environment which can be used to complement current KBDDSs and to ease the difficulty of the conceptualising task. The intention of modelling such situations is to accept that in some cases the human intellect cannot be surpassed or imitated, but aspects of the system can be designed as so to facilitate the conceptualising task. For example, the meanings of objects (relations) in the specified UoD, which are interpreted and provided by users according to the background culture and convention of the organisation, are difficult to encode or predescribe in a unified, internal form and cannot be reduced to a set of rules. But the design of the intelligence component can assist designers in some respects, and can therefore indirectly reduce the confusion surrounding the identification and interpretation of objects in the unbound, dynamic reality.

In summary, the aims of this research are:

1. To explore the nature of the reality-conceptualising (conceptual data modelling) task and examine the current status of developed KBDDSs,

together with a consideration of a complementary theoretical knowledge model.

2. To construct a knowledge model, Dual-Base Knowledge Model (DBKM), by taking account of the essence of the conceptual data modelling world and the problem-solving behaviour of experts.
3. To analyse the process model of the constructed knowledge model.
4. To develop a theoretical architecture, including the theory and elaborated process mechanism, which can be used as a rational basis for creating the symbiosis of humans and computers, and as a valid basis for complementing the environment provided by current KBDDSs.
5. To test the architecture by means of the simulated example in order to clarify the function of the DBKM.

1.2 Method of Knowledge Modelling

For the purpose of constructing a knowledge model which can be used to complement the environment provided by the developed KBDDSs, the current status of KBDDSs is first examined according to the nature of the reality-conceptualising (conceptual data modelling) task to highlight their advantages and disadvantages in helping the designers in the conceptual data modelling work.

In this study, the contents of the conceptual schema are viewed as the main body of the specific, concrete experience (knowledge). In order to reuse the relevant

stored schemata, two important inherent properties of such reuse are considered. These are: the phenomena of multiple interpretations and polymorphic objects, and the implications of recalling and structuring the schemata. In order to propose a unified theoretical architecture to handle these in a systematic view, the theories of conceptual knowledge in the psychological realm, including the notions of the scene-schema mental model, the exemplar view of object concepts and the Case-Based Reasoning (CBR) paradigm, are adopted and formalised as the groundwork of the knowledge model construction.

Based on the premise of the relevance of the mental world and mental model, a knowledge model called Dual-Base Knowledge Model (DBKM) is proposed to handle the multiple interpretations and polymorphic objects by means of the notion of the exemplar view of object concepts. Regarding the mental world as being crucial and of primary interest as the mental model for handling the nature of the reality conceptualisation, the DBKM includes two bases, the concept base (derived from the mental world) and the case base (derived from the mental model), whose aim is to complement each other synergistically in order to reuse the relevant conceptual schemata by means of the CBR paradigm and thus assist users (designers) to comprehend the problem, recognise certain requirements, and elicit other relevant information from external knowledge sources.

In order to place the DBKM on more formal ground, some notions from set theory are used to underpin the framework of the DBKM. Two relations are defined to express the relationship between cases. These are Proper Subset-Supset relation

(abbreviated as sub-super relation) and Overlapped relation (abbreviated as overlp relation), which can be of help in reusing the relevant old cases and facilitating the acquisition of new cases. Two functions are defined to co-ordinate the relevance of the concept and case base. These are the surjective function: the correspondence between the concepts (relations) of a corresponding part of the concept base and the entities (relationships) of a case, and the bijective function: the correspondence between the concept and case base, which provide a valid basis of the interplay of the two bases. The above definitions and the intrinsic properties of the relations in the DBKM can be used and transformed into different forms of meta-knowledge to control the old case retrieval and new case store processes.

Two process mechanisms, the case retrieval and knowledge accumulation mechanisms, which are elaborated from the general process cycle of the CBR paradigm, are developed in the DBKM environment. By means of the meta-knowledge embedded into the case retrieval mechanism, the relevant stored schemata can be recalled from the memory pool, and by means of the meta-knowledge embedded into the knowledge accumulation mechanism, the newly gained knowledge can be added into the knowledge repository at the appropriate place for retrieval in the next design session. For clarification of how the relevant cases can be retrieved from the DBKM and the new gained knowledge can be retained in the DBKM, a simulated example is given to provide a comprehensive understanding the function of the DBKM.

The theoretical architecture of the DBKM is evaluated by comparing it with (1) the frameworks of developed KBDDSs and other CBR systems (from the knowledge-based systems view) and (2) the knowledge models in current KBDDSs and rule-based data modelling approaches (from the knowledge modelling view). Thus, the limitations of the proposed DBKM are discussed.

The contributions of this architecture are twofold: (1) for the knowledge-based database design aid environment; and (2) for the constructed DBKM itself. Finally, there is a brief discussion of the need to extend this theoretical architecture by accommodating the dynamic and functional aspects of the UoD.

1.3 Overview of Chapters

In Chapter 2, the framework of this research is discussed from two perspectives: (1) the current status of the developed KBDDSs considered in terms of knowledge formality, philosophical perspectives and cognitive process; (2) the research motivation inspired by the importance and the hermeneutic view of the conceptual data modelling, and the inherent properties of the conceptual schemata reuse.

In Chapter 3, the foundation of the DBKM is described by presenting the essential background knowledge: the perspective on conceptual knowledge, the exemplar view of the theory of concepts, and the paradigm of CBR.

In Chapter 4, the framework of the DBKM is introduced in detail including the cognitive aspects of the DBKM, the structures (contents and contexts) of the concept and case base, the relevance of these two bases, the formalisation of the dual-base knowledge structure, and the intrinsic properties of the relations in the DBKM for retrieving the relevant concrete experiences.

In Chapter 5, the process cycle of the DBKM is illustrated by the two mechanisms: the case retrieval mechanism and the knowledge accumulation mechanism. The case retrieval mechanism consists of three core recalling processes whose function is to recall the relevant experiences from the case base. These are the first-pass focus process, the recalling cycle and the adapt sub-process. The knowledge accumulation mechanism is the storing sub-process which retains the new gained knowledge into the DBKM.

In Chapter 6, the simulated example of how the relevant concrete experiences are retrieved and the new knowledge is stored is exemplified, providing a comprehensive view for understanding the function of the DBKM. The distinctions between DBKM architecture and the frameworks of current KBDDSs and other CBR systems (from the knowledge-based system view), and between the DBKM and those knowledge models in current KBDDSs and rule-based data modelling approaches (from the knowledge modelling view) are discussed, and the limitations of the current DBKM are evaluated.

In Chapter 7, the main contributions of this research are discussed from two points of view: (1) for the knowledge-based database design aid environment - constructing a theoretical model, analysing an elaborated process model, and providing an intelligent component; (2) for the DBKM itself - making the knowledge meaningful, theoretically improving the beliefs underlying the fact-based KBDDS approaches, and providing a hermeneutic environment. Finally, there are suggestions as to how future research can extend the current status of DBKM by accommodating the dynamic and functional aspects of the UoD; and three key availability issues are identified: the layered user interface, the parallel computing algorithms, and the client/server architecture.

Chapter2

The Current Status of KBDDSs and the Research Motivation

The main problem areas of conceptual database design are the weak-theory domain and the open-world domain. That is, the relations between (among) objects are uncertain (weak theory domain) [Porter et al., 1990, Aamodt, 1990a]; and the category of the object is open and unbounded, the causal relationships and facts relevant to a given problem may not be fully known, and the proposed solution for problems is adequate rather than correct (open-world domain) [Hinrichs, 1992]. This chapter investigates the limitations of current KBDDSs, the hermeneutic situation of the conceptualising task, and the inherent properties of the schemata reuse to provide a rational framework of knowledge modelling construction. In section 2.1 the current status of KBDDSs is examined in terms of three aspects: knowledge formality, philosophical perspectives and cognitive process. In section 2.2 the factors motivating this research are explained in terms of the importance and hermeneutic view of the conceptualising task, and properties of the conceptual schemata reuse.

2.1 The Current Status of KBDDSs

The environment of KBDDS is intended to provide effective automated assistance for the complex and time-consuming task of database design. Several prototype KBDDSs have been built to apply AI techniques to assist or replace database designers. To highlight and facilitate the previous efforts of current KBDDSs, the conceptual database design may be separated into two sub-phases: object identification and interpretation and syntactical processing, as illustrated in Figure 2.1. According to this differentiation, the emphasis of each sub-phase of the conceptualising work in the KBDDS environment is defined as follows: (1) in the object identification and interpretation sub-phase, the emphasis is on how to provide an environment which can assist designers to capture a holistic comprehension of the specific UoD and to facilitate the projection through the systematic correspondence with ill-defined objects (relations) in semantic-rich reality and well-defined entities (relationships) in the specific UoD; and (2) in the syntactical processing sub-phase, the emphasis is on how to provide an environment including constructs and mechanisms based on a semantic data mode for structuring and constraining the data items. Then the conceptualisation is viewed as an iterative process of object identification and interpretation from the semantic-rich reality to the specific part of reality and manipulating syntactics based on the theory of a conceptual data model until the conceptual schema is finished in the KBDDS environment. For instance, the simulated processes of the conceptual design of the I^2S and VCS systems are exemplified respectively to

illustrate the distinction between the object identification and interpretation sub-phase and the syntactical processing sub-phase (the data source is extracted from Storey, 1993).

A simulated conceptual design session of I^2S[1]

I^2S : Enter the name of your database:

User : Acme Engineering

(Specify the UoD)

I^2S : What is the domain of 'Acme engineering' ?

Please choose one from the following:

1) literature 2) company 3) school ...

User : 2

(Object Identification and Interpretation Sub-Phase: Relationships)

I^2S : Now let's start designing the conceptual schema for Acme Engineering.

User : I want to know which employees work on which projects.

I^2S : Do you want to know that employee-1 works on project-1?

User : Yes.

I^2S : Please enter the next information requirement.

User : I want to know which managers manage which employees.

I^2S : Do you want to know that manager-1 manages employee-1?

---

[1] The current version of the system deals only with relationships and cannot handle entities or their attributes.

User : Yes.

•

I^2S : Please enter the next information requirement.

User : No more.

I^2S : Wait a moment, let's discuss the following ...

(At this point, the system would bring up any remaining to be resolved)

•

(Syntactical Processing Sub-Phase)

I^2S : Ok, I start designing the conceptual schema. Wait a moment, please.

Designed conceptual schema is as follows:

work-on: [employee, project, start_date, end_date[2]].

manage: [manager, employee].

A simulated conceptual design session of VCS

VCS : Enter your name please.

User : Ian.

(Specify the UoD)

VCS : Hello ian, for what do you want wish to design a database?

User : company

(Object Identification and Interpretation Sub-Phase: Entities)

VCS : Please enter the name of an entity.

User : employee.

VCS : What are the attributes for 'employee'?

---

[2] 'start_date' and 'end_date' are relationship attributes and might be obtained from a user's statement such as 'I want to know about the start_date and end_date for employees working on the project'. It is not clear whether I^2S could deal with such a requirement.

User : emp# name address dept.

VCS : Any more attributes for 'employee'?

User : no.

VCS : Can any of these attributes have more than one value for any occurrence of 'employee'?

The user is allowed to ask the system why it is asking certain questions.

User : Why?

VCS : If an attribute can have more than one value, then these values may be instances of the same 'thing'. That 'thing' should be an entity ...

VCS : Enter an attribute or combination of attributes which could be used to identify UNIQUELY an instance of the entity 'employee'. I will call this attribute (or group of attributes) a key.

User : emp#.

VCS : Is there any other way that an instance of the entity 'employee' could be identified ?

User : name address.

VCS : Enter next entity.

User : manager.

VCS : What are the attributes for 'manager'?

User : emp# name address dept yrs-experience employee.

The system helps the user determine whether something should be an entity, an attributes or a relationship.

VCS : I already know that 'employee' is an entity. Does this mean that you will need to know 'employee' before you will be able to identify UNIQUELY an instance of 'employee' ?

User : no.

VCS : Then you really telling me that a relationship exists between 'manager' and 'employee'. we will go to that in a minute.

(Syntactical Processing Sub-Phase)

An initial set of entities and their candidate keys is obtained[3].

Person : [SSN, name, address, date-of-birth].

Person : [NAME, ADDRESS, ssn, date-of-birth].

Employee : [EMP#, name, address, dept, phone].

Employee : [NAME, ADDRESS, emp#, dept, phone].

Manager : [EMP#, name, address, dept, yrs-experience].

Manager : [NAME, ADDRESS, emp#, dept, yrs-experience].

Project : [PROJECT_ID, description].

(Object Identification and Interpretation Sub-Phase: Relationships)

Relationships :

VCS : Enter a relationship of the form A Verb Phrase B.

User : employees work-on projects.

VCS : Is there at least one 'project' for every 'employee'?

User : yes.

VCS : Is there at most 'project' for every 'employee'?

User : no.

VCS : Is there at least one 'employee' for every 'project'?

User : yes.

VCS : Is there at most one 'employee' for every 'project'?

User : no.

(Syntactical Processing Sub-Phase)

From this the system infers that the min/max values are employee(1,) and project(1,*)*

•

---

[3] Candidate keys are capitalised. For example, two candidate keys are identified for Person : [SSN] and [NAME, ADDRESS].

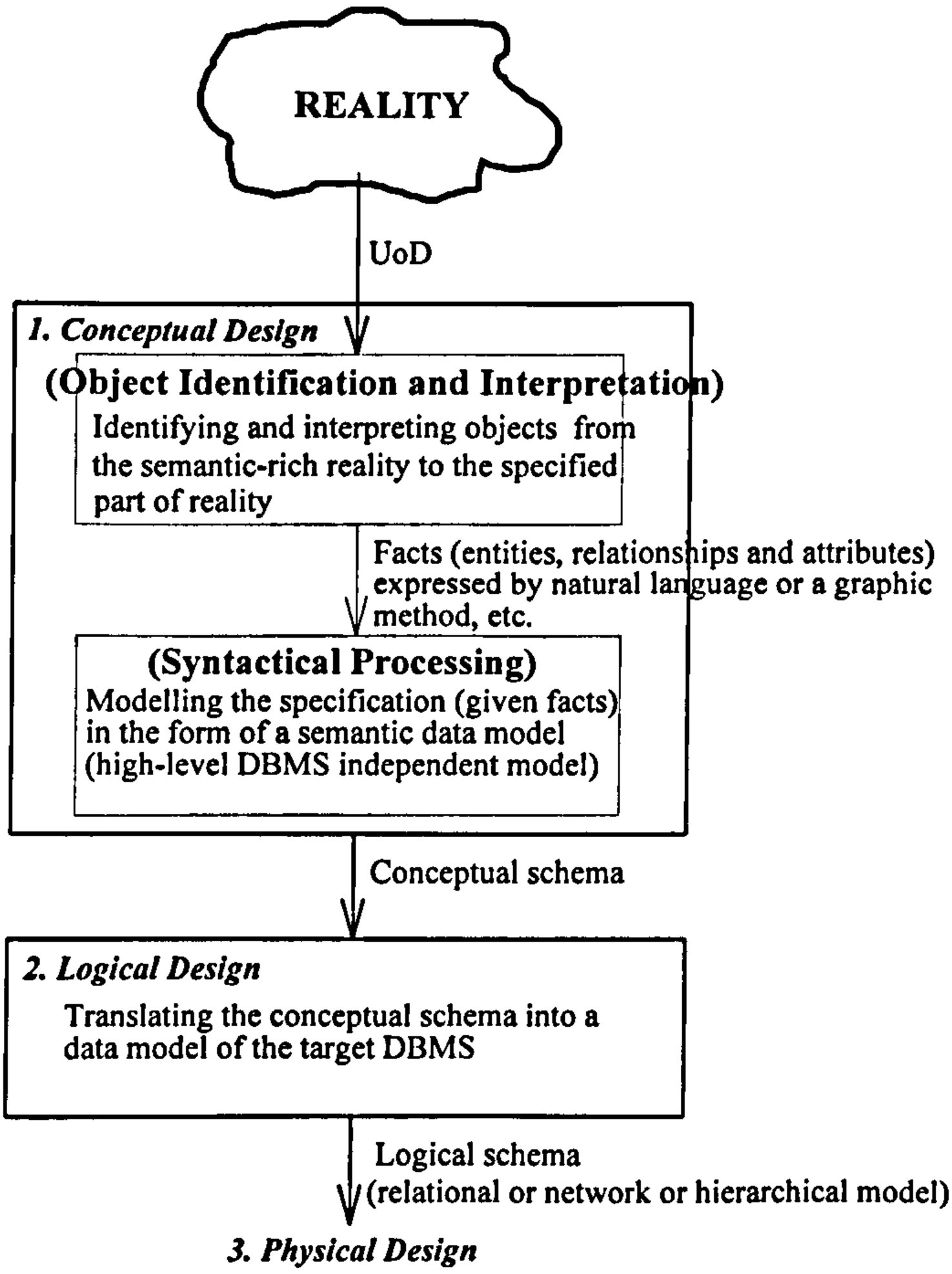


Figure 2.1 : The database design process of the KBDDS (the conceptual and logical design phases)

2.1.1 Knowledge Formality in the Syntactical Process Mode

According to the above definition, all the current KBDDSs surveyed by Lloyd-Williams and Beynon-Davies [1992], Storey [1993], Storey and Goldstein [1993] and Storey et al. [1996] have been classified into the syntactical processing sub-phase or the logical design phase depicted in Table 2.1. The classification of the current KBDDSs into a particular sub-phase of the conceptual design is based on

the particular types of task, such as object identification and interpretation or syntactical processing, which are focused upon no matter what semantic data models they use and what semantic integrity constraints they are embedded in to structure and constrain the data items. Two important characteristics of current KBDDSs are recognised by this classification: the logical approach and a domain-independence environment. The logical approach is adopted by all of the developed KBDDSs in their knowledge-base construction, whatever the formalisms they use, such as propositional or first-order predicate calculus, and whatever the architectures they adopt, such as logic-based or rule-based, and no matter by what structure the facts (entities, relationships and attributes) are expressed, such as a flat fact base, semantic nets or frames. Basically, the property of domain-independence of those KBDDSs is a consequence of using the logical approach by virtue of: (1) the *sound*, deductive inference procedure, which means that the conceptual schema is satisfied by the given facts that can be viewed as an interpretation of the specific UoD in the syntactical processing sub-phase, and the logical schema is satisfied by the finished conceptual schema that can be regarded as another form of an interpretation of the specific UoD in the logical design; in other words, the finished conceptual and logical schema are all satisfied by the given facts; and (2) The CWA (Closed-World Assumption) that depends on a syntactic feature of a set of beliefs; namely, whether a positive ground literal (fact) can be derived. Under this assumption, every schema is finished by a *complete* deductive inference procedure. The CWA is based on the conventions of how people draw a conclusion from observed facts. For instance, if I want to know whether there is an inter-city train from Coventry to Manchester at 13:00,

but the timetable mentions no train to Manchester leaving at 13:00, then I will conclude that there is indeed no train to Manchester at 13:00. This convention is most efficient and convenient when the 'positive facts' are few in number compared with the 'negative facts'. A database design agent (a human designer or KBDDS) using the CWA will want to conceptualise the domain in a way that matches this expectation. So, while the objects have been identified and interpreted (i.e. the facts are given), the KBDDSs, under the sound and complete deductive inference procedure, can use the syntactical rules for deriving the implicit facts that are logically implied by the given facts, and then the finished conceptual and logical schema are all satisfied by the given facts.

Although using the abstracted, generalised rules can make the current KBDDSs become domain-independent environments, the lack of ability to ease the difficulty of object identification and interpretation resulting from the unbound, dynamic reality is also attributed to these kinds of rules, which influences the attitudes of thinking and feeling about the reality and the types of knowledge, thus resulting in the use of the Rule-Based Reasoning (RBR) paradigm as a main pattern for constructing the KBDDSs.

Table 2.1 : Classification of the current KBDDSs

System	Knowledge Representation	Conceptual Object Identification and Interpretation	Design Syntactical Processing	Logical Design	Domain Independence
I^2S	Rules & Frames	✗	✓	✓	✓
VCS	Rules	✗	✓	✓	✓
CARS	Rules	✗	✓	✓	✓
Modeller	Rules	✗	✓	✓	✓
EDDS	Rules	✗	✓	✓	✓
CAERM	Rules	✗	✓	✓	✓
AVIS	Rules & Algorithms	✗	✓	✗	✓
SECSI	Rules & Semantic Nets	✗	✓	✓	✓
ER-Translator	Rules & Semantic Nets	✗	✗	✓	✓
GESDD	Rules	✗	✓	✓	✓
CHRIS	Rules	✗	✓	✓	✓
EXIS	Logic & Semantic Nets	✗	✓	✗	✓
OICSI	Rules & Semantic Nets	✗	✓	✓	✓
E^2R	Logic	✗	✓	✓	✓
Gambit	Logic	✗	✓	✓	✓
TSER	Rules & Algorithms	✗	✓	✓	✓
PROEX	Rules & Semantic Nets	✗	✓	✓	✓

NONE	YES
✗	✓

2.1.2 Philosophical Perspectives on the Rule-Based Reasoning (RBR) Paradigm

Analysing the philosophical stances of the current paradigm affords a systematic comprehension of the strengths and weaknesses of the developed KBDDSs and provides the rationale of enquiry for complementing the current method. A useful starting point is to follow Klein and Hirschheim [1987] in characterising a paradigm into two sets of assumptions: ontology (beliefs concerning the nature of the UoD) and epistemology (beliefs concerning the nature of knowledge of the UoD and how it is acquired).

The ontological assumption of the RBR paradigm in KBDDSs is that facts are given "out there". That is, the specific UoD comprises immutable entities and structures, and affirms the existence of a domain of identifiable nonchanging entities, and of nonchanging relationships between these entities.

The epistemological assumptions of the RBR paradigm in KBDDSs involve using rules as the source of primary knowledge and explaining observable phenomena through the identification of causal relationships expressed by the general rules: heuristic principles and model rules (theory of data models, such as the conceptual and logical data models).

The above philosophical assumptions are *true* in the syntactical processing sub-phase and the logical design phase because the users' requirements and the

conceptual data schema are given and immutable, and the theories of the data model can be translated into a number of rules. However, the assumptions become *unacceptable* in the object identification and interpretation sub-phase. Here the reality which the designers understand and with which they interact is chaotic for the following reasons:

(1) There are an infinitely large number of objects that exist and might be included in a schema, but only some objects are relevant and suitable for choice. For example, a database of a business organisation does not normally include those things called 'HOSPITAL' and holds no stored information about them since they have no relevance to the company's interests. If, however the company enrols its employees in a private medical insurance scheme, then decisions may need to be made which require information about local hospitals, and hospitals then become relevant to the organisation's interests [Lewis, 1994].

(2) The same referent (object) might be represented by different primitives contained by the conceptual data model, such as the entity or relationship construct in the ER model. For example, the 'COLOUR' object is an *attribute* for the entity 'CAR'. However, if an application is concerned with building pieces of furniture, and particularly with their colouring as a process, then 'COLOUR' may well become an *entity,* with the attributes *name*, *colour_code*, *required_number_of_paints*, *rust_protection*. The 'MARRIAGE' object is an *attribute* called *marriage marital status* for the entity 'PERSON' in the personnel system, but might be an *entity* in the marriage registry system [Batini, 1992].

(3) The properties (attributes) and roles of an entity might be different. For example, the properties and roles of the 'AUTHOR' entity might be different in the library and bookstore systems. In the library system, the attributes might be *name* and *profession*, and the roles it is assigned might be *written* with the 'BOOK' entity and *published* with the 'PAPER' entity. In the bookstore system, the attributes might be *name*, *sex*, *age*, *tel_no*, *fax_no*, and the roles might be *live-in* with the 'CITY' entity and *written* with the 'BOOK' entity.

(4) The degrees of relationships of a given set of entities are expressed in diverse ways. For example, given three entities, there are three (C_2^3) ways of defining binary relationships and one (C_3^3) ternary relationship is possible.

(5) The cardinalities of a relationship between (among) the same entities might be different. For example, the cardinality of the [*participated*] relationship between 'EMPLOYEE' and 'PROJECT' entities might vary in different organisations. The cardinality (1,1) of the 'EMPLOYEE' side, expressing the idea that each employee participates in exactly one project in one company, might be turned into (1,n) in another company, expressing the idea that each employee participates in at least one project.

Considerations (4) and (5) mean that the possible configurations of relationships from a given set of entities are numerous, and that they are 'non-polynomial' while the number of entities increases [Batra and Zanakis, 1994].

As a consequence of the essence of reality in the object identification and interpretation stage, no analytical rules can be used for the automated transfer of

this mapping. The desire to emphasise the formulation of syntactical rules that can be used to draw logical conclusions makes the current KBDDSs try in vain to facilitate object identification and interpretation and then to project this knowing to the conceptualising work. This limitation is inherent in the approach they use - the RBR paradigm. The desire to help designers identify and interpret the meanings of the objects (relations) in the specific UoD from the semantic-rich reality is beyond their abilities.

2.1.3 Cognitive Process in the Facet-Based Model

Typically, current KBDDSs are all based on the facet-based model[4] [Batra and Davis, 1992; Storey, 1993]. For example, SECSI [Bouzeghoub et al., 1985], EDDS [Choobineh et al., 1988] and GESDD [Dogac et al., 1989] all use the logical approach to capture specific facets, such as identifying binary relationships based on forms or specifying dependency information inferred from interviewing the users.

In the conceptual design phase, experienced designers use two types of knowledge for handling the task of conceptualisation. One is the formal knowledge of the semantic data model they use in order to form the specification in a coherent way (corresponding to the syntactical processing sub-phase in the KBDDS

---

[4] The term 'facets' refers to the entities, relationships qualified by degree and cardinality, attributes (identifiers and descriptors) and categories [Batra and Davis, 1992].

environments); and the other is their experiences, which enable them to recognise similar problems and to treat them by analogy in order to focus the relevant information of a specific UoD from the semantic-rich reality (corresponding to the object identification and interpretation sub-phase in the KBDDS environment). An expert designer has extensive application experiences. When carrying out the conceptual design, experts are first able to relate the current task to previous examples and respond without thinking to set a situation based upon the compiled but concrete knowledge - the past experiences and cases that resemble the current problem - in order to constrain the problem space and further meet the highly relevant user-requirements, and then use the general, context-free or heuristic rules to model the requirement specifications into the formal form [Johnson, 1983; Storey et al., 1995].

Unfortunately, current KBDDSs are all constructed to automate the syntactical checking by means of rules translated by the semantic data model and they leave the effort of reusing the experiences to the users by means of user interfaces, such as the natural language and/or a graphic interface. The absence of the ability to reuse the previous specific experiences, like database design experts, limits the availability of current KBDDSs [Lloyd-Williams and Beynon-Davies, 1992; Storey and Goldstein, 1993] and obviously makes them inadequate in reaching the level of typical experts.

The result of using this method means that the current KBDDSs are restricted to the representation level rather than in the enterprise and recognition levels, like

human experts, for example by developing a holistic comprehension of the problem domain and using domain-related experiences to understand the application [Batra and Davis, 1992]. This finding is consistent with the work of following scholars: (1) Chase and Simon [1973], Johnson et al.[1981], Chi et al.[1981] and Schvaneveldt[1985] reported that experts are able to perceive a more global picture and to encode large amounts of information into meaningful chunks that can be evoked from memory as they encounter similar situations; (2) Fairley [1985], Adelson and Soloway [1985], Guindon [1990] and Batra and Davis [1992] suggested that experts rely upon the well-understood, correct experiences to draw a picture of the completed high-level software design for comprehending the problem and for retrieving further information to solve the familiar problems as soon as the experts begin their work.

Using the syntactical rules, by virtue of the RBR paradigm, to check automatically the semantic integrity constrained by the given facts makes the current KBDDSs 'local' environments that are only concerned about specific facets. The limits to provide a 'global' environment that can be used to help the designers capture a comprehensible picture of the specific UoD, first by means of reusing the well understood, correct experiences and then doing the right thing, and in consequence the behaviour of current KBDDSs is unlikely to match the typical expert.

The above limitations caused by the RBR paradigm have led KBDDS practitioners to rethink how to reuse the useful experiences from solved problems rather than to refire the rules embedded in the systems. For example, Mannila

[1996] presents an idea for reusing the relations (tables) from the previous logical design stage; Castano and Pernici [1966] suggest a new direction for the conceptual schemata reuse, for example by using the domain knowledge and adopting the case-based reasoning paradigm; and Chang et al. [1996, 1997] propose a concept-case knowledge model based on the cognitive theories of concepts and the case-based reasoning paradigm for conceptual database design.

2.2 The Research Motivation

In section 2.1, the advantages and disadvantages of supporting the conceptual data modelling work in current KBDDSs were highlighted. To further motivate this research, in this section the importance and the hermeneutic view of the object identification and interpretation are described, and the inherent properties of conceptual schemata reuse are considered.

2.2.1 The Importance of Object Identification and Interpretation

The process of object identification and interpretation stage in conceptual database design involves interviewing the user-groups and analysing the existing documents or systems, i.e. specifying the data requirements which the organisation needs to carry out its business. Nowadays, influenced by the evidence put forward by some researchers of IS development, it is recognised that

the quality of a database system depends greatly on the accuracy of the requirements specification. For example, the potential cumulative impact of errors in the requirements specification is substantial, and contributes to the rising costs of information development [Mizuno, 1983]: the 56 per cent of all bugs detected can be traced to errors made during the requirements stage [Tavolato et al., 1984].

Therefore, the object identification and interpretation sub-phase is vital to the success of the whole process of database design. The quality of object identification and interpretation in conceptual design is a necessary condition for a successful database design and makes a strong contribution to the later stage of database design, implementation and maintenance.

Although the importance of object identification and interpretation have been acknowledged, there is something of a sleight of hand in the literature on database design-aiding environments. That is, much work has led to proposing more expressive and richer semantic integrity-constraining *syntactical tools*, such as current KBDDSs, ERwin^{R}, DesignAidII^{TM}[5]. The provision of an environment to alleviate the confusion of identification and interpretation objects (relations) from semantic-rich reality has not been considered because of its complexity.

---

[5] For further information, see Reiner [1992:430-435] and Storey et al [1996:30-31].

2.2.2 The Hermeneutic View of the Conceptualising Task

Conceptual database design involves two major activities: (1) identifying and interpreting objects (relations) from the semantic-rich reality; (2) modelling the cohesive object-structure of the specific UoD. In the modelling stage, the nature of the UoD, while the objects (relations) have been identified and interpreted, is comprised of an immutable object-structure (objects and relations), and thus the fact-based modelling approaches (such as the ER model) can be applied to this situation in terms of those objects and relations. An important question which arises in the object identification and interpretation stage is: how can the designers and users set a clear mapping from the ill-defined objects (relations) in the semantic-rich reality to the well-defined entities (relationships) in the specified UoD without hesitation and confusion. In current KBDDSs, the problem raised by the above issue is pushed aside, not because it is uninteresting, but because it is too difficult and open-ended. By concentrating on the formulation of systematic rules that can be used to draw logical conclusions, the practitioners of current KBDDSs can develop clear technical results whose validity can be judged in terms of internal coherence and consistency.

Current KBDDSs share the three beliefs underlying the fact-based data modelling approaches. These are the beliefs in (1) the objective nature of the UoD; (2) the factual descriptive nature of information; and (3) the consensual role of the UoD [Hirschheim et al., 1995]. These beliefs are true for the agents of current KBDDSs, but false for human designers. The reality encountered by the designers

is one of chaos filled with much uncertainty and many unknowns, and is not given but needs interpretation and communication.

In order to resolve the confusion surrounding the conceptualising task, this study adopts the notion of hermeneutics [Winograd and Flores, 1987; Patula, 1992; Hirschheim et al., 1995] to investigate a knowledge model: an environment of conceptual data schemata reuse, which can be used to complement current KBDDSs. The notion of reusing the relevant schemata, which can be of help in pre-understanding the working domain (e.g. the personnel or library UoD), is based on the premise that a similar object-structure pattern will recur in a specific UoD, although the contents and roles of objects in this structure under the organisational constraints might be different.

In the object identification and interpretation stage, designers read the organisation and its intended users as a text in order to make an interpretation that will provide the basis for a conceptual data schema design, i.e. the database design in this stage is a hermeneutic task. Therefore the designers find themselves in the hermeneutic situation with regard to the tradition that they are trying to understand, and do not stand outside it. This act of conceptualisation is possible only within the horizon given by the preunderstanding of the designer. Thus, the process of conceptual database design corresponds to the hermeneutic cycle in which the meaning of an individual object is contextual, depending on the moment of conceptualisation and the horizon brought to it by the designer. In this study, the proposed knowledge model, constructed by the exemplar view of the object concept and the Case-

Based Reasoning (CBR) paradigm, provides this hermeneutic environment in which the stored concrete experiences (cases) are regarded as a number of possible worlds (cases). The designer accesses the relevant case in order to narrow and give structure to his horizon. Using this horizon designer can help to pre-understand the problem domain and ease the communication with users, and thus an intersubjective dialogue between the designers and users can be established effectively. By means of a number of the intersubjective dialogue processes, an effective constitution and change of the designer horizon can be accomplished, the meaning of the text can be constituted, and thus the object-structure of the specified UoD can be modelled.

2.2.3 The Inherent Properties of Conceptual Schemata Reuse

The finished schemata can be viewed as knowledge representation schemata [Hirschheim et al., 1995]. This means that conceptual data modelling is the process of justifying the beliefs about the conceptual data structure in the individual mental world until all the members in the specific UoD achieve consensus. Thus, the contents of a finished conceptual schema express the justified belief which is shared by the individuals in the specified world. In terms of reusing this knowledge, two issues are crucial: (1) the multiple interpretations and polymorphic objects (relations) in the realities; (2) the implications of recalling and structuring the concrete experiences in the knowledge repository. These two issues are discussed below.

2.2.3.1 Multiple interpretations and polymorphic objects (relations) in the reality

In conceptual database design, every relevant world object is interpreted as an entity constant, and every relevant world relation is interpreted as a relationship constant. An interpretation in a UoD is a mapping between the ill-defined objects (relations) in the semantic-rich reality and the well-defined entities (relationships) in the specific part of reality. For example, a part of an interpretation in the LIBRARY domain is shown in Figure 2.2.

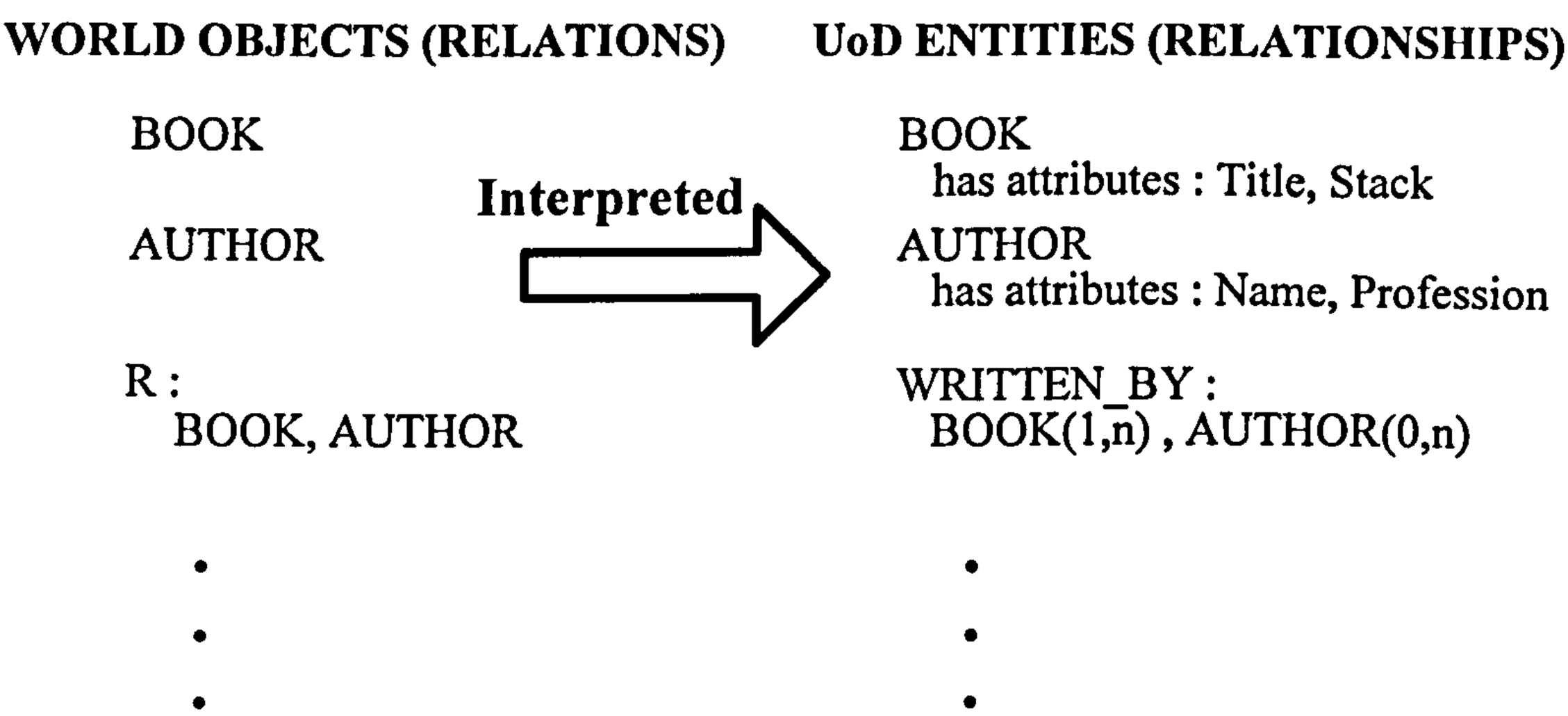


Figure 2.2 : A part of an interpretation in the LIBRARY domain (UoD)

The conceptualising task is not independent of use. The process of creating such a conceptual schema is not value-free and the resulting schema is not neutral, i.e. it will be greatly affected by the values, beliefs and expectations of those creating it, not only by 'facts'. The constraints on conceptualisations are pragmatic, and the

contents of the conceptualisation in a specific UoD are determined in great measure by the use to which they are put. A specific UoD can be modelled by different conceptual schemata (i.e. existing multiple interpretations) and still be satisfied.

Consequently, an object can be identified and interpreted in multiple forms. The object identification and interpretation sub-phase are inherently processes which are realised in our interaction with the environment and depend upon the frame of reference. That is, the meanings of objects (relations) in the specified UoD can not be encoded or predescribed in a unified, internal form, and this is also why they cannot be reduced to laws. As such, 'an observer can comment on them, but they can not be reduced to single point-of-view, object description' [Clancey, 1991: 392], and the nature of reality, including polymorphic objects (relations), is hard to fulfil with the semantics in a first-order predicate calculus (FOPC), where the basic semantics of terms is defined through strict deductive inference (modus ponens) only [Birnbaum, 1991; Aamodt and Nygard, 1995].

The consideration of object identification and interpretation from ill-defined objects (relations) to well-defined entities (relationships) is typically based upon their intended uses influenced by the conventions and culture of an organisation. No analytical rules can be used to automate the transfer of this mapping. In order to handle the complexity of the circumstances that arise from the essence of the conceptualisation, in this research the systematic correspondence with ill-defined

objects (relations) in the semantic-rich reality and well-defined entities (relationships) in the specific UoD is set up to facilitate this projection work.

2.2.3.2 Implication of recalling and structuring concrete experiences in the knowledge repository

According to the substantial evidence provided by psychological investigations and discussed in section 2.1.3, expert designers not only have many experiences in their minds but also use them for solving similar encountered problems. Every experience can be regarded as a potential interpretation of similar problems. So, while designing a new conceptual database, experts first recall the previous experiences that are similar in this working domain without thinking and then get a picture of this domain for the purpose of further adaptation. There are many interesting findings based on some common experiments in psychology which investigate the analogical reasoning of people. That is, even though human designers have many experiences, they are not always good at identifying relevant past experiences and often fail to remember potentially useful experiences [Gick and Holyoak, 1980, 1983; Gentner, 1989; Domeshek et al., 1994a]. For example, Gick and Holyoak [1980, 1983] reported in their experimental findings how subjects used the analogical 'Attack-Dispersion' story (in which a general captured a centrally located fortress by having small units of soldiers attack simultaneously along multiple roads) to solve the 'radiation problem', which involves finding a way for a doctor to use X rays to destroy a stomach tumour without damaging the surrounding healthy tissues. This finding demonstrated that

75 per cent of those subjects produced the convergence solution to the problem when a hint to use the military story was given, while only 20 per cent of subjects reached the solution without that explicit hint. This situation not only occurs in cross-domain examples, but also in the same domain. For example, students seldom notice analogies between problems presented in different chapters of their geometry textbooks [Holyoak et al., 1987].

As a result of the observations of experts and people's behaviour, the issues of the reuse of memorable experiences and of concrete cases have to be fully considered in order to complement current KBDDS environments. Two issues are involved in experience reuse: these are design by reuse and design for reuse. How to recall useful concrete experiences from the memory pool and how to retain the newly gained knowledge into the knowledge repository are the main concerns of the former issue; and the latter issue is concerned with how to structure knowledge, including representation, segmentation and indexing.

For the purpose of reusing concrete experiences like experts and avoiding the weakness of human designers, in this research a unified theoretical architecture of the schemata reuse environment regarding these two issues as two sides of the same coin is proposed to address these implications.

Chapter 3

The Background Knowledge of the Dual-Base Knowledge Model (DBKM)

The DBKM is intended to provide an environment that can be used to overcome the characteristic confusion in the object identification and interpretation stage by recalling previous relevant, concrete experiences. In order to achieve this aim, two elements are built into the proposed knowledge model, DBKM, for assisting the designers to *identify* and *interpret* the related objects and the relations between them in the specified UoD. One of the elements relates to the systematic, corresponding surroundings that are used to handle the relevance of ill-defined objects (relations) and well-defined entities (relationships); and the other element is the reasoning mechanism which can be used to recall similar experiences. The former is resolved by the exemplar view of object-concept, and the latter is achieved by the CBR paradigm. In terms of their origins, these two bases of the DBKM both stem from the theories of conceptual knowledge in the psychological realm. In order to understand the origins and subsequent development of the theory used to construct the DBKM, the view of conceptual knowledge is first presented in section 3.1, and then the two foundations are discussed in the next two sections.

3.1 The Perspective on Conceptual Knowledge

During the past three decades, many psychologists have proposed and demonstrated how concepts are associated in our minds and influence our understanding. In their respects, conceptual knowledge[6], one kind of the meaning-based knowledge representations[7], is constructed by concepts and links, and coheres in specifiable ways rather than being formed by a number of particular, unconnected facts. Accordingly it can be used to categorise and understand objects by virtue of the class-inclusion relations among the concepts, and to perceive and comprehend something by virtue of the spatial relations of objects in some situations and the temporal relations of activities in a common routine.

The trend of this thought was initiated and encouraged by the hypothetical model of semantic memory[8] proposed by Quillian [1968]. The postulated model, which

---

[6] This stands in contrast to propositional representation, but they are both types of meaning-based knowledge representation. The major distinction between them is the form of knowledge they handle. For example, the smallest unit of knowledge of propositional representations is a 'proposition' borrowed from logic and linguistics that can stand as a separate assertion; while the basic unit of knowledge of conceptual knowledge is a 'concept', which may be an object type, such as bird, apple, house or an event type, such as going to the theatre, buying the ticket, seeing a movie, etc. For further details, see Anderson [1995].

[7] The fundamental characteristic of such representations is that they involve some significant abstraction away from the perceptual detail in contrast with the perception-based representations which tend to preserve much of the structure of the original perceptual experience [Anderson, 1995].

[8] Semantic memory refers to the storage and utilisation of knowledge about words and concepts, their properties and interrelations [Tulving, 1972]. Thus, research with the semantic network, the

consists of concepts and links, suggested that human beings store information about various categories[9], such as canary, ostrich, fish, in a network structure, and this served as a basis for demonstrating the effect of how the structure affects people's understanding. This was examined by some experiments into how subjects judged whether a statement was true or false [Collins and Quillian, 1969] and into the relationship between the organisation of facts in semantic memory and their retrieval times [Conrad, 1972]. An example of a semantic network is presented in Figure 3.1.

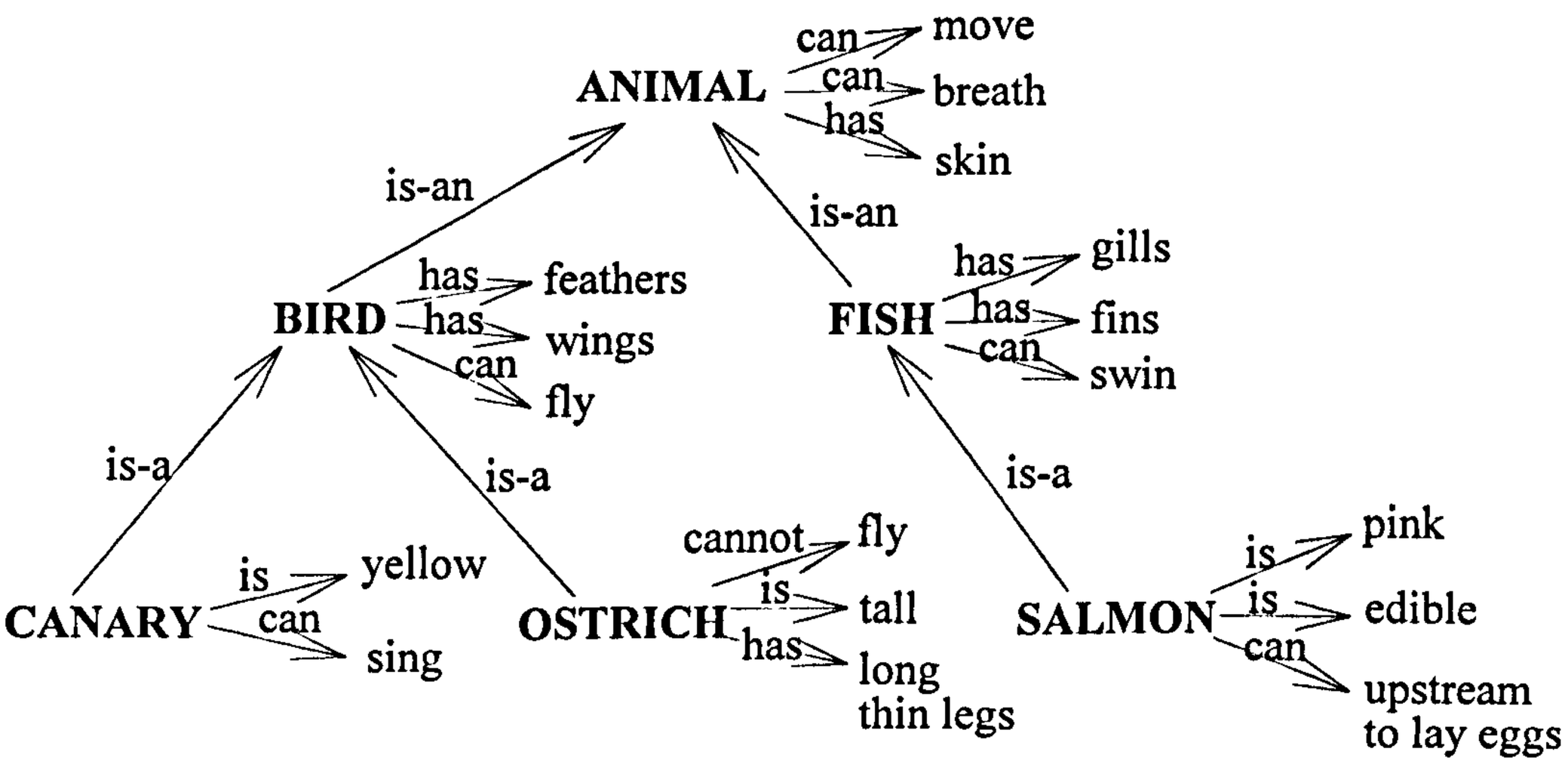


Figure 3.1 : A postulated semantic network structure
Source : Collins and Quillian [1969]

---

views of object concepts and the framework of higher-level structure: P-schema are all within the scope of semantic memory.

[9] A category is distinct from an object concept, which is defined as a mental representation of a category. For example, the category 'dog' includes all real and imaginary dogs in the world, and the object concept 'dog' is an idea that allows people to categorise various animals as dogs [Howard, 1987].

Following these experiments and investigations, two research directions proceeded almost in parallel in the analysis of conceptual knowledge. One concerns the internal structure of mental categories, and the other is concerned with the schematic structure of interconnected concepts called the schema[10] (to avoid confusion, the term 'schema' is used in relation to database design, and the 'P-schema' is used here for the psychological case). In section 3.2, the evolved representations of the mental category are contrasted.

The notion of P-schema, proposed to enhance the insufficient semantic networks, postulates that there is another structure for storing larger interrelated chunks of knowledge by means of the spatial or temporal relations. The most important contribution of the proposal is that it illustrates the general theoretical framework of how people perceive and comprehend something by instantiating the appropriate schematic structures. That is, the typical objects or activities in one situation will first be inferred unless the others are noticed explicitly[11] [Rumelhart,

---

[10] The notion of schema is an old one but was largely ignored in early psychological studies with the exception of the work of the British psychologist, Frederic Bartlett. For him, the schema is defined as 'an active organisation of past reactions, or of past experiences, which must always be supported to be operating in any well-adapted organic response' [1932: 201] and is viewed as a generic, unconscious mental structure. Thus, the memory is organised around schemata containing summaries of familiar stories and situations rather than a fixed set of remembered episodes. People perceive and comprehend something by means of instantiating the related standard schemata which are guided by the individual's 'attitudes' (feelings and affects), i.e. the process of remembering is a total reconstruction by means of incorporating incoming episodic information into the corresponding generic mental structure.

[11] This categorisation is by means of the p-schemata in subjects' mind that they allow subjects to place the default value of this situation.

1980; Brewer and Treyens,1981]. For example, in the *living room* scene[12], the spatial, related objects, such as window, sofa, coffee table, TV and fireplace, are typically those that we expect to see. In the *restaurant* script[13], as sketched in Table 3.1, the optional sequence of the temporal, prescribed activities will be predicted, e.g. entering the restaurant and sitting down, picking up the menu and ordering food, eating the food, and paying the bill and leaving. The different concerns of the two mental structures exemplified above have greatly influenced the implementation of the CBR paradigm resulting in its two distinct branches. This paradigm has evolved and elaborated the P-schema framework from two perspectives.

---

[12] The scene is one of the P-schema types [Mandler, 1984; Howard,1987]. It specifies the spatial relations of objects we are likely to see in certain situations. There are two characteristics of the structure of scene schemata where people are asked to list the parts of ordinary scenes, such as a park, living room, school, restaurant: (1) most of the listed things are 'basic-level objects' [Tversky and Hemenway, 1983], but include little about the descriptive details of what the basic-level objects look like (Mandler and Stein, 1974; Mandler and Ritchey, 1977], i.e. subjects do much better at remembering *what* objects were in the scene; (2) the vertical information of scenes is remembered better than horizontal information suggesting that the more invariant knowledge about spatial relations is contained in the vertical information of scenes; for example, the information that windows and pictures are on walls which are above sofas and a TV in a living room scene is more specific than the horizontal information, e.g. that a sofa is to the right or the left of a TV.

[13] The script is an event P-schema. It postulates that part of our knowledge is organised around hundreds of stereotypic situations with temporal activities such as visiting a dentist, riding a bus, attending a lecture, grocery shopping [Schank and Abelson, 1977]. The function of the script is that it can serve as valuable basis for predicting missing information and for correcting errors in information [Bower et al., 1979].

One relates to the memory type it addresses: The memory model of the CBR paradigm can be used to accommodate episodic memory[14] (the specific, conceptual knowledge structures - cases) and semantic memory (the general, conceptual knowledge structures - schemata) rather than to accommodate the semantic memory only by the framework of the P-schema. The task of problem-solving in the CBR paradigm involves recalling the similar, specific experiences (cases) by means of using the general knowledge (schemata) for biasing the search, guiding the adaptation, choosing the indexes and matching the cases, etc. rather than by instantiating the standard, abstracted schemata by means of incorporating the incoming episodic information in the framework of the P-schema.

The other perspective relates to the issues of processing and storage structures. In the CBR paradigm, the process model, based on the postulate of dynamic memory regarding the structure for processing the new experiences as the same structure for organising the memory [Schank, 1982], provides more specific, processing algorithms and explicit storage structures rather than being just a rough outline for processing issues in those P-schemata, which completely ignore the issues of storage structure.

---

[14] Episodic memory refers to the storage and retrieval of concrete, temporally dated, spatially located and personally experienced events or episodes [Tulving, 1972] and is a specialised subsystem of semantic memory [Tulving, 1985]. For example, a particular experience of cycling, such as 'Remember that time we went on a picnic by bike .....', represented in episodic memory, can be viewed as an example of the explicit knowledge of how to ride a bike represented in the semantic memory.

In section 3.3, the two types of memory model and the general cycle of the CBR paradigm will be discussed.

Table 3.1 : A restaurant script
Source : Anderson [1995]

Script : RESTAURANT

Scene 1 : Entering

Customer enters restaurant
Customer looks for table
Customer decides where to sit
Customer goes to table
Customer sits down

Scene 2 : Ordering

Customer picks up menu
Customer looks at menu
Customer decides on food
Customer signals waitress
Waitress comes to table
Customer orders food
Waitress goes to cook
Waitress gives food order to cook
Cook prepares food

Scene 3 : Eating

Cook gives food to waitress
Waitress brings food to customer
Customer eats food

Scene 4 : Exiting

Waitress writes bill
Waitress goes over to customer
Waitress gives bill to customer
Customer gives tip to waitress
Customer goes to cashier
Customer gives money to cashier
Customer leaves restaurant

3.2 The Internal Structure of Object Concepts

> Without concepts, mental life would be chaotic. If we perceived each entity as unique, we would be overwhelmed by the sheer diversity of what we experience and unable to remember more than a minute fraction of what we encounter. And if each individual entity needed a distinct name, our language would be staggeringly complex and communication virtually impossible [Smith and Medin, 1981: 1].

In the object identification and interpretation sub-phase, one of the tasks facing database designers is to decide how to translate the object concepts[15] and the relations between them from the semantic-rich reality into their corresponding, well-defined entities and relationships in a specific UoD, i.e. the contents and roles of object concepts are the main bodies with which the designers are concerned at this stage. Notwithstanding the concern in psychological investigations that the theories and mental representations of object concepts are beyond the database design boundary, the empirical findings concerning how people characterise the nature of object concepts provide richer insights into how to co-ordinate the relevance of multiple entities (relationships) that are interpreted from the same object concept(relation) that is the basis of the DBKM.

---

[15] The object concepts stand in contrast to the event concepts, such as visiting a doctor, eating out, grocery shopping which pertain to time. Thus, the object concepts used in the conceptual database design phase represent some physical things at some locations or cultures in real or imaginary space, such as 'book', 'person', 'teacher', 'author', 'publication' in the library or bookstore UoD; 'customer', 'credit_card', 'shipping_information' in the bank or trade-company UoD; and 'officer', 'family', 'reward/punishment', 'education', 'career' in the military personnel UoD.

'Concepts are critical for perceiving, remembering, talking, and thinking about objects and events in the world' [Smith and Medin, 1981: 1]. The psychological studies of how people categorise objects into particular types lead to the question of how object concepts are represented in the human mind. Numerous experiments based on various theories have been set up to explore the possible views of the object concept. Smith and Medin [1981] distinguish between the 'classical', 'probabilistic' and 'exemplar' views. The major principle of *the classical view* is that it assumes every concept has a set of *defining* features that are singly necessary and jointly sufficient to classify examples of a category. For example, the concept 'bird' consists of defining features such as 'lays eggs', 'has feathers' and 'can fly'. To determine whether a penguin, parrot or bat is a bird, people must assess whether they have all the defining features. This view is sometimes called the 'rule model' [Millward, 1980]. That is, acquiring a concept can be linked to learning a rule such as 'All animals which [lay eggs], [have feathers] and [can fly] are birds', and applying it to classify objects.

The probabilistic view presumes that a concept is defined not by the defining, but only by a set of *characteristic* features or *mean values* on several dimensions of a category[16], i.e. modal features that objects tend to have but need not have, or salient dimensions which are likely to be non-necessary ones. For example, as shown in Figures 3.2 and 3.3, the 'bird' concept might include features such as

---

[16] Although the holistic approach is the third notion of the probabilistic view, it lacks a formal theory of how to represent and classify a category [Smith and Medin, 1981]. In this research, we dispense with this approach and go on to two other approaches, the featural and dimensional approaches.

'has wings', 'has feathers', 'can fly', 'can sing', or dimension-mean pairs such as 'animacy-animate', 'size-s', 'ferocity-f'. Each feature (dimension-mean pair) has a weight attached to it that specifies its importance. So in this view the concepts are learned by adding weighted features, or by adding weighted dimensions and then calculating the means on relevant dimensions to an evolving concept definition as they are encountered in new instances[17]. Categorising a specific bird, like an ostrich, is not based on whether it possesses all these features (dimensions), but by checking those features (dimensions) and seeing whether the weighted sum of those is reached or the metric distance (only for the dimensional case) is less than the certain thresholds[18]. According to this principle, this view is also called the 'prototype view', where the mental category is defined by some measure of the central tendency[19] of some instances, which can be a highly typical instance or an idealisation. Objects are then categorised as exemplars or non-exemplars by reference to this prototype, i.e. the membership of a category is graded by the

---

[17] The weight and mean value are to be taken as subjective rather than objective. They are subjective because they may be biased towards particular values experienced during a critical acquisition period instead of being based on an unbiased estimate of how frequently each feature (dimension) and dimension mean occurs in instances [Smith and Medin, 1981].

[18] Basically, the specific models of the weighted sum or metric distance computation can be classified into two types depending on whether the contrast concepts are involved in this categorising. For example, the spreading activation model [Quillian, 1968; Collins and Loftus, 1975] and the simple distance model [Rips et al., 1973; Rosch et al., 1976a] do not include the contrast concepts in their computation. However, the cue validities [Rosch and Mervis, 1975] and comparative distance model [Reed, 1972; Palmer, 1978] do consider the influence of contrasting concepts.

[19] The measure of central tendency can be either a modal value for the characteristic case, or a mean (average) value for the dimensional case.

degree of prototypicality rather than in an homogeneous status resulting from the restricted conditions of the classic view.

Robin	Chicken	Bird	Animal
1.0 moves	1.0 moves	1.0 moves	1.0 moves
1.0 winged	1.0 winged	1.0 winged	0.7 walks
1.0 feathered	1.0 feathered	1.0 feathered	0.5 large size
1.0 flies	1.0 walks	0.8 flies	
0.9 sings	0.7 medium size	0.6 sings	
0.7 small size		0.5 small size	

Figure 3.2 : The featural approach of the probabilistic view
Source : Smith and Medin [1981]

Robin	Chicken	Bird
1.0 animacy -animate	1.0 animacy -animate	1.0 animacy -animate
0.7 size - s_R	0.7 size - s_C	0.5 size - s_B
0.4 ferocity - f_R	0.4 ferocity - f_C	0.5 ferocity - f_B

lowercase letters (for example, s, f) designate specific values on a dimension

Figure 3.3 : The dimensional approach of the probabilistic view
Source : Smith and Medin [1981]

Neither represented as an abstracted set of defining features nor defined by some measure of central tendency, concepts postulated in *the exemplar view* are represented by their *exemplars*, which are either instances or subsets, rather than by an abstract summary[20]. For example, as depicted in Figure 3.4, the 'bird' concept might be represented with subsets such as robin, bluejay, sparrow and canary, where they are likely to be represented by their own exemplars or by abstracted

---

[20] Although the representation of a concept also involves some abstraction, the properties of a concept are implicit in its exemplars instead of being explicit in its abstracted definition.

definition, and/or instances such as a specific canary - 'fluffy'. Obtaining a concept involves simply storing exemplars of its category members. Two possible models have been suggested for determining which exemplars are to be stored: the proximity model [Reed, 1972], according to which a concept is represented by every unique example which is encountered; and the best-examples model [Mervis, 1980], according to which a concept is represented by the stored N typical exemplars based on two principles: the family resemblance and contrast set principles[21]. To decide whether a specific entity is of a particular kind, the categorisation is by reference to the stored exemplars. Two different selection criteria are used to determine which exemplar(s) is (are) retrieved for the purpose of comparison. One is by retrieving all the exemplars of a concept to check whether the specific entity is a *sufficient match* to at least one exemplar, which implies that the same information is always accessed as the category membership is being determined. The other, based upon the context model[22] [Medin and

---

[21] That is, the better examples (exemplars) are those that have large numbers of features that are widely shared with other category members while at the same time having few features that are commonly presented in a number of other similar categories. This view of the mental representation of a category is adopted according to some findings about the phenomena of the family resemblance scores and the typicality ratings, e.g. (1) a few instances usually share the highest score [Rosch and Mervis, 1975]; (2) two or more instances attain comparable maximal ratings [Rips et al., 1973; Rosch, 1975a]; (3) the more plausible image of the superordinate concept, for example, the concept of 'animal' seems to require multiple exemplars of bird, mammal and fish [Smith and Medin, 1981].

[22] The criterion of the classification judgement is based on the following two assumptions [Smith and Medin, 1981] : (1) An entity X is an instance or a subset of category Y if and only if X retrieves a criterial number of exemplars of mental category of Y before retrieving a criterial number of exemplars from any contrasting concept. (2) The probability that entity X retrieves any specific exemplar is a direct function of the similarity of X and that exemplar. For example, if we

Schaffer, 1978], uses the degree of *similarity* between a given entity and the stored exemplars for determining which exemplar is retrieved for the purpose of comparison, which implies that the different exemplars in the concept are accessed for different categorising entities.

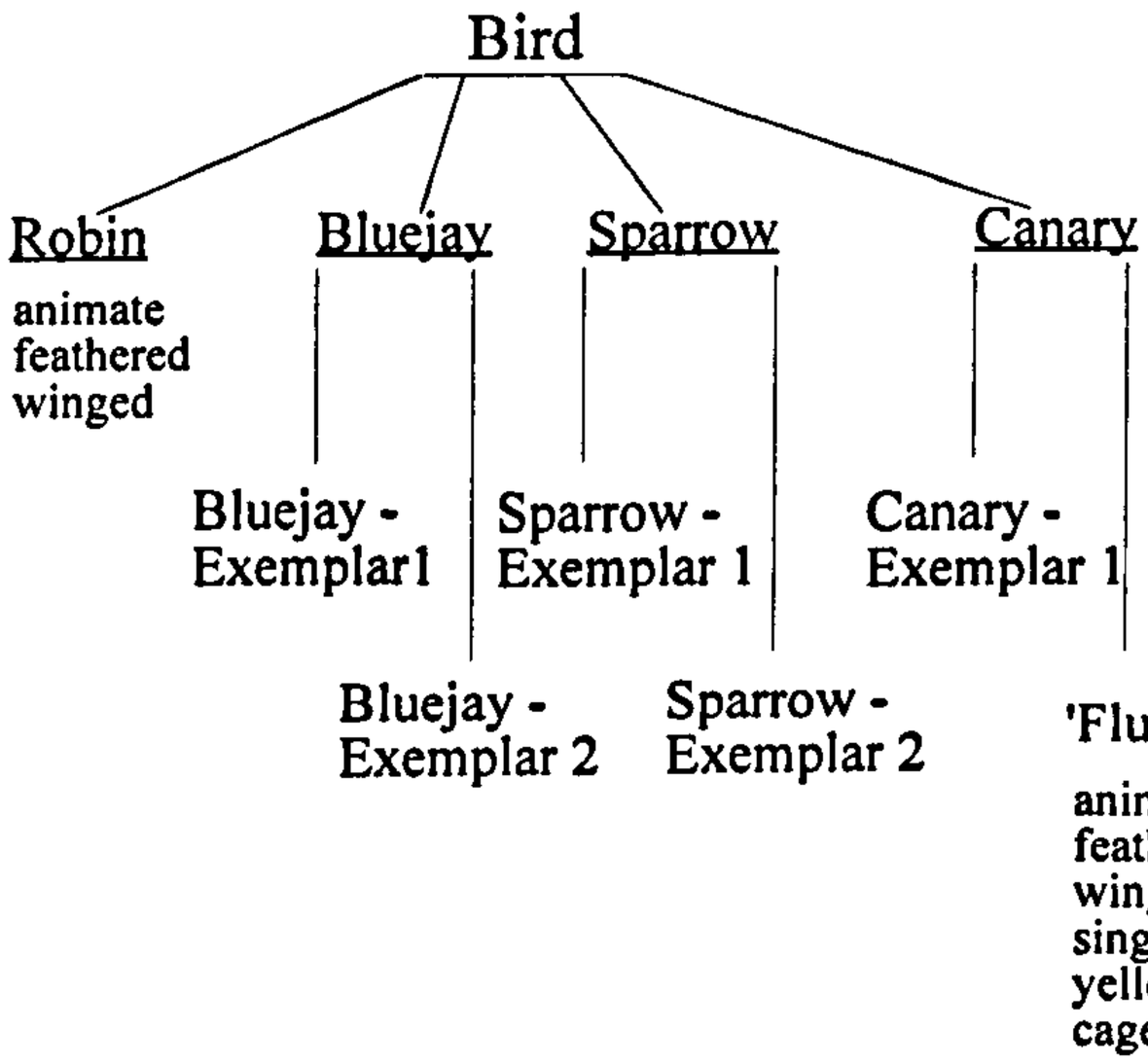


Figure 3.4 : An exemplar representation of the 'Bird' concept
Source : Smith and Medin [1981]

Although each of the three views of mental category discussed above suggests the content of representation of concepts, the answer to the question of how concepts are mentally represented is always confused by different views of the nature of a theory of concepts and depends upon the intended use of the theory [Smith and Medin, 1981; Mervis and Rosch, 1981; Cohen and Murphy, 1984].

---

use the feature descriptions, then the similarity between test instance and an exemplar is a direct measure of shared features.

3.3 The Paradigm of Case-Based Reasoning (CBR)

> The CBR approach is based on two tenets about the nature of the world. The first tenet is that the world is regular: similar problems have similar solutions. Consequently, solutions for similar prior problems are a useful starting point for new problem-solving. The second tenet is that the types of problems an agent encounters to recur. Consequently, future problems are likely to be similar to current problems. When the two tenets hold, it is worthwhile to remember and reuse prior experience: case-based reasoning is an effective reasoning strategy [Leake, 1996: 1].

In the object identification and interpretation sub-phase, another challenge facing database designers is how to *identify* the object concepts and the relations between them in the specified UoD from the semantic-rich reality. Reasoning using cases (analogs) is a natural process for people when there is much uncertainty and during early learning [Ross, 1989; Kolodner, 1993; Malek and Labbi, 1995]. Being used under a variety of circumstances for problem-solving and decision making, Case-Based Reasoning (CBR) is able to utilise the specific knowledge of previously experienced concrete problem situations and is a very fast method for learning comparing to classical models like back-propagation. It has been shown to be particularly effective reasoning strategy in complex, dynamically changing situations [Klein and Calderwood, 1988; Kolodner, 1993; Leake, 1996]. In this research, the CBR paradigm is regarded as another basis of the DBKM that

provides a better way to ease the identification problem by virtue of its intrinsic character.

3.3.1 The Types of Case-Based Reasoning Memory Model

The hallmark of CBR is that it solves new problems not by first principles, but by retrieving and modifying previously useful experiences with similar problems. Although many systems and applications have been implemented according to this paradigm, they can be distinguished into two major groups depending on what type of *concepts* their memory model is concerned with. The first is based on *event-type memory*. For example, CYRUS (Computerised Yale Retrieval and Updating System) [Kolodner, 983a; 1983b; 1984], the first computational model of CBR to implement Schank's dynamic memory, uses *event-memory organisation packets* (E-MOPs), which are conceptual categories of events, to organise general knowledge - the content frames (schemata) representing normative information about a class of events - and the indexed specific knowledge (episodes)[23], which are the specialisation of those general knowledge structures. The sample structure of the event memory model of CYRUS is exemplified in Figure 3.5: While the contextual information (the diplomatic meeting E-MOP) has been specified by means of querying the users[24], the

---

[23] Indices of an event (episode) represent salient features of the event and these features single out ways in which an event differs from other events and from the normative information of a class of event.

[24] Basically, CYRUS is a question-answering system.

retrieval process operates by traversing the appropriate indices to locate the target event. If, for instance, 'Begin' cannot be located, then the memory of a diplomatic meeting between Vance and Begin cannot be accessed[25]. CYRUS could be regarded as a milestone of the CBR paradigm because the event memory model has subsequently influenced several other CBR systems in different domains such as a resource-dispute resolver, MEDIATOR [Simpson, 1985]; a recipe planner, CHEF [Hammond, 1989]; a meal designer, JULIA [Hinrichs, 1992].

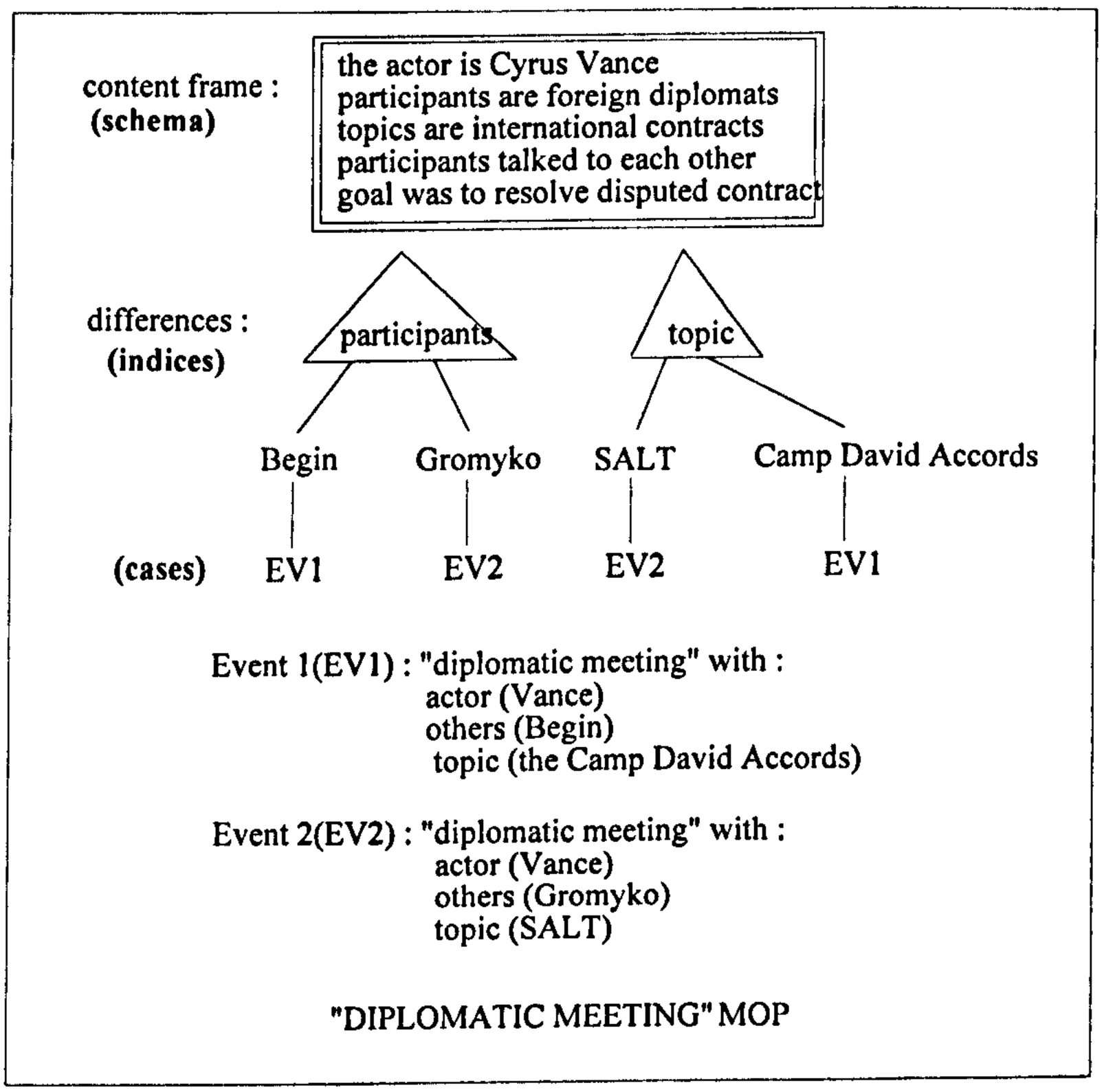


Figure 3.5 : The event-memory organisation packet (E-MOP) of CYRUS
Source : Kolodner [1984]

---

[25] Retrieval processes in CYRUS are fairly complex and it is not intended to describe them here. For further details, see Kolodner [1984] book: Chapters 3 to 6.

The second type of memory is *object-type memory*. Two different formations of the conceptual categories of objects have been used to structure the specific knowledge as instances (exemplars) of those structures. The first platform of knowledge structure is constructed mainly by the notion of the exemplar view of concept. For example, PROTOS [Bareiss, 1989; Porter et al., 1990], the first computational model of CBR based on the theories of object concepts[26] ignored by Schank's theory, classifies a given case by using similar stored exemplars of the candidate categories and learns by retaining exemplars and explanations under expert guidance[27]. As a unified solution of concept representation, classification, and learning, the exemplar-based memory structure of PROTOS is augmented by explanations and indices. An explanation associated with an exemplar feature represents the domain knowledge, thus making a category more coherent by the equivalencies among the features of the exemplars of a category. For example, as illustrated in Figure 3.6, *armrests* are relevant to *chairs* because '*armrests* enable *holds (person)*, which is the function of *chairs*' and *metal* is equivalent to *wood* because both make chairs. Four kinds of semantic links (indices) which are reminding, censors[28] (negative reminding), prototypicality and difference links, in

---

[26] Those theories include: the exemplar view of concepts [Smith and Medin, 1981]; the theory of concept or category coherence [Medin and Smith, 1984; Murphy and Medin, 1985]; the notion of polymorphous concepts [Hampton, 1979]; and the principle of family resemblance [Rosch and Mervis, 1975].

[27] The PROTOS is an apprentice-expert system, i.e. PROTOS gains new knowledge (new exemplars or explanations) by interacting with the expert to obtain the correct classification and an explanation of why it is correct. In Bareiss [1989], PROTOS was applied to the domain of audiological (hearing) disorders.

[28] The censor link consists of two types: nonabsolute (its value is expressed as a negative numeric strength value such as -0.3, -0.5, etc.) and absolute links (its value is expressed as $-\infty$).

PROTOS memory allow it to select exemplars for comparison, as shown in Figure 3.7. Basically, the classification procedure of PROTOS is that of feature-driven comparison. That is, the features of the input case are used to generate a list of all plausible categories, for instance the *chair* category (the greatest strength of reminding), by means of the reminding and censor links[29]. Consequently, the most highly typical exemplar, for instance the *chair1*, of that category will be chosen to match[30] by means of the prototypicality links. If the case partially matches *chair1*, but has an unmatched feature, for instance the *leg (4)*, *chair2* is suggested by the difference links from *chair1* to *chair2*[31].

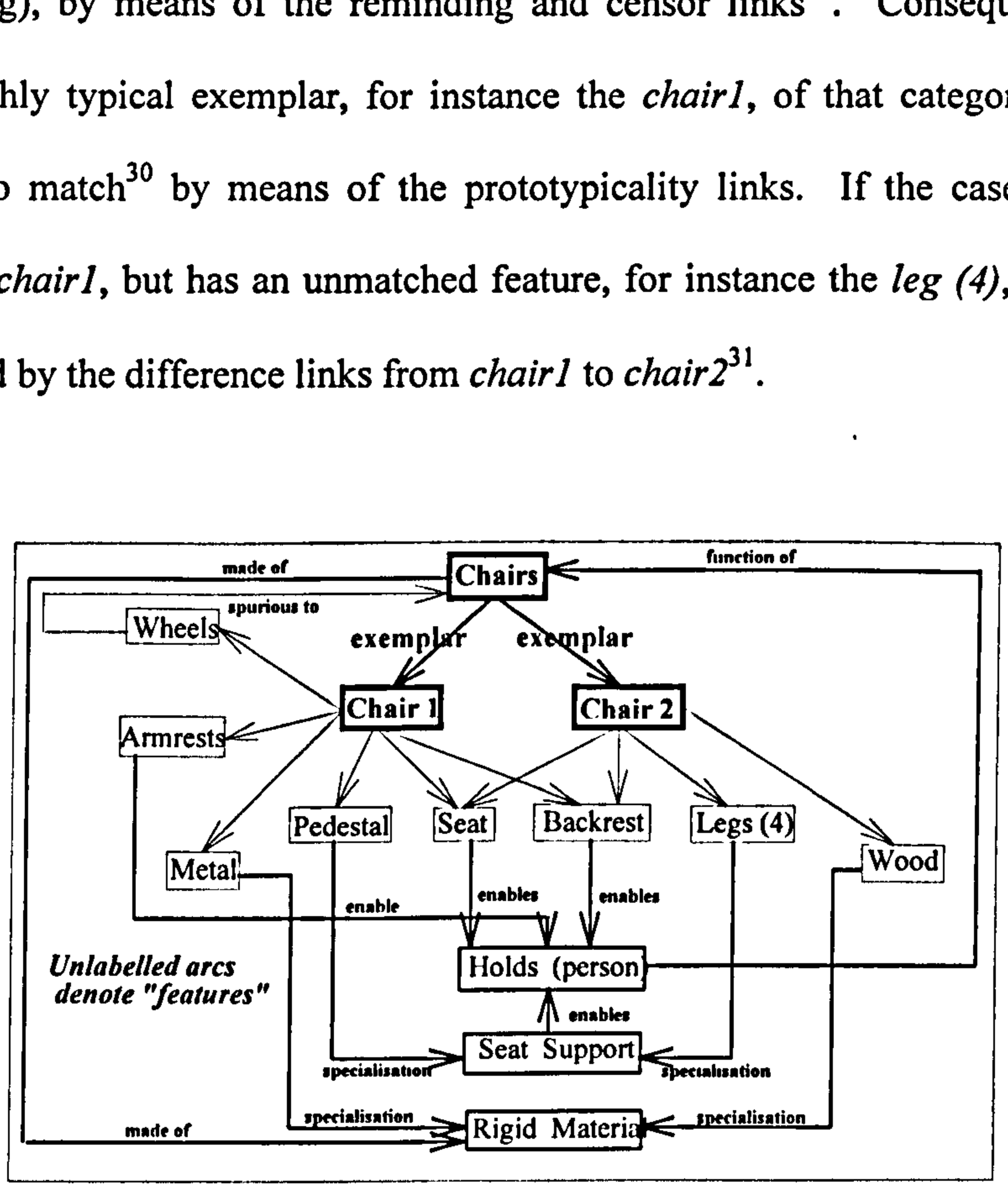


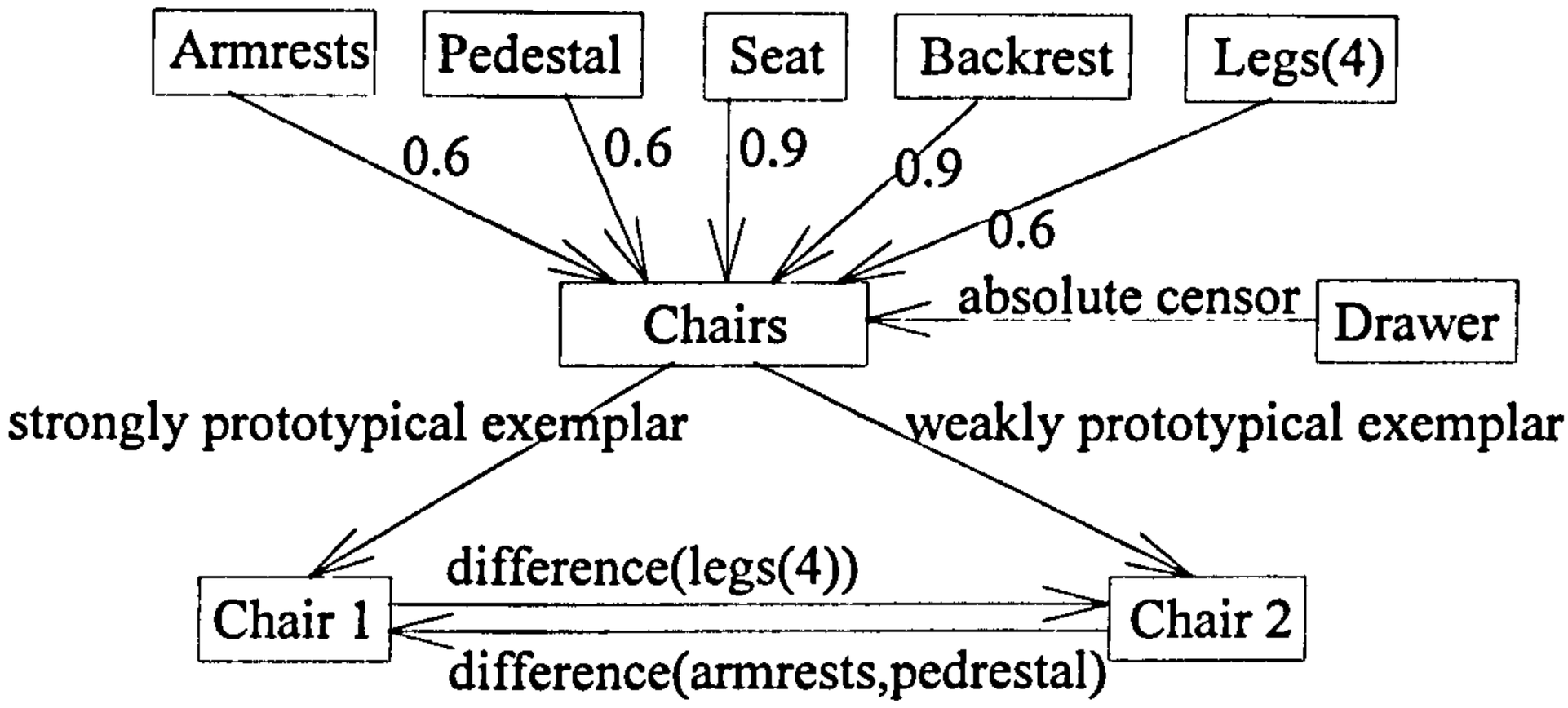
Figure 3.6 : A sample category structure of the memory model of PROTOS
Source : Porter et al. [1990]

---

[29] For example, in the domain of furniture, the features of a new case might be associated with the categories chair, table, bench, stool.

[30] For example, the robin, canary and bluebird have strong prototypicality links with bird. By contrast, the owl and penguin have weak prototypicality links with bird.

[31] The classification process of PROTOS is very complex. For example, the selected candidates are likely to be exemplars as well as categories. Other exemplars in the same or different categories are used to match when the matching is unsuccessful (judged by experts), etc. For further detailed considerations, see Bareiss [1989].



Unlabelled arrows denote remindings (the number indicates strength)

Figure 3.7 : The semantic links for 'chairs' category structure
Source : Bareiss [1989]

The second platform of knowledge structure is constructed mainly by the general conceptual structure, *scene*, representing some spatial, object concepts and invariant relations among them in some domain; and specific conceptual structures, *cases*, as specifications of a part of the things (relations) that is relevant for some goal in the domain scene. For example, CREEK (Case-based Reasoning through Extensive Explicit Knowledge) architecture [Aamodt, 1989; 1990b; 1994a], a unified conceptual knowledge fundament, solves the diagnosis tasks by recalling previous, similar experiences (cases) based on the more general scene knowledge structure, including goal-related, task-related and domain object concepts, and relations among them. A partial conceptual structure of a domain of *car starting problem* represented by CREEK is illustrated in Figure 3.8: the *general scene* structure, which consists of the relevant concepts including *goal-related concepts* such as find-treatment, find-fault; *task-related concepts* such as diagnosis, diagnosis-hypothesis, engine-test, starter-motor, etc.; *domain concepts*

such as vehicle, car, electrical-system, etc.; and the invariant relations among them such as *wheel*, *electrical-systems*, *fuel-system* and *engine* are all **[part-of]** *car*, *diagnosis* is **[described-in]** a *diagnostic-case* being a **[subclass-of]** *case*, and the *car fault* has **[subclass-of]** the *fuel-system-fault*, *electrical-fault* and *engine-fault* etc. *Case#54* (shown in Figure 3.9), consisting of part of the instantiating objects (relations), is hooked into the scene structure by means of some domain relation. For example, *car#1* is an **[instance]** of *car*; *starter-motor-turns* is a **[status-of]** *starter-motor*, *not-confirmed* is a **[possible-status-of]** *diagnostic-hypothesis*, etc. As an explanation-driven CBR approach, the CREEK architecture uses this densely coupled semantic network as a pool of explanations for further selecting the best retrieved case[32], modifying the solution provided by the best matching case and deciding whether the new case is retained.

---

[32] Basically, while the input case description is given, the process of retrieving the similar stored cases involves activating the related concepts, forming them as findings, and then using these findings as indices to retrieve a set of similar cases whose matching strength is above a certain threshold by means of the relevant relations in the semantic network. For further details, see Aamodt [1994a].

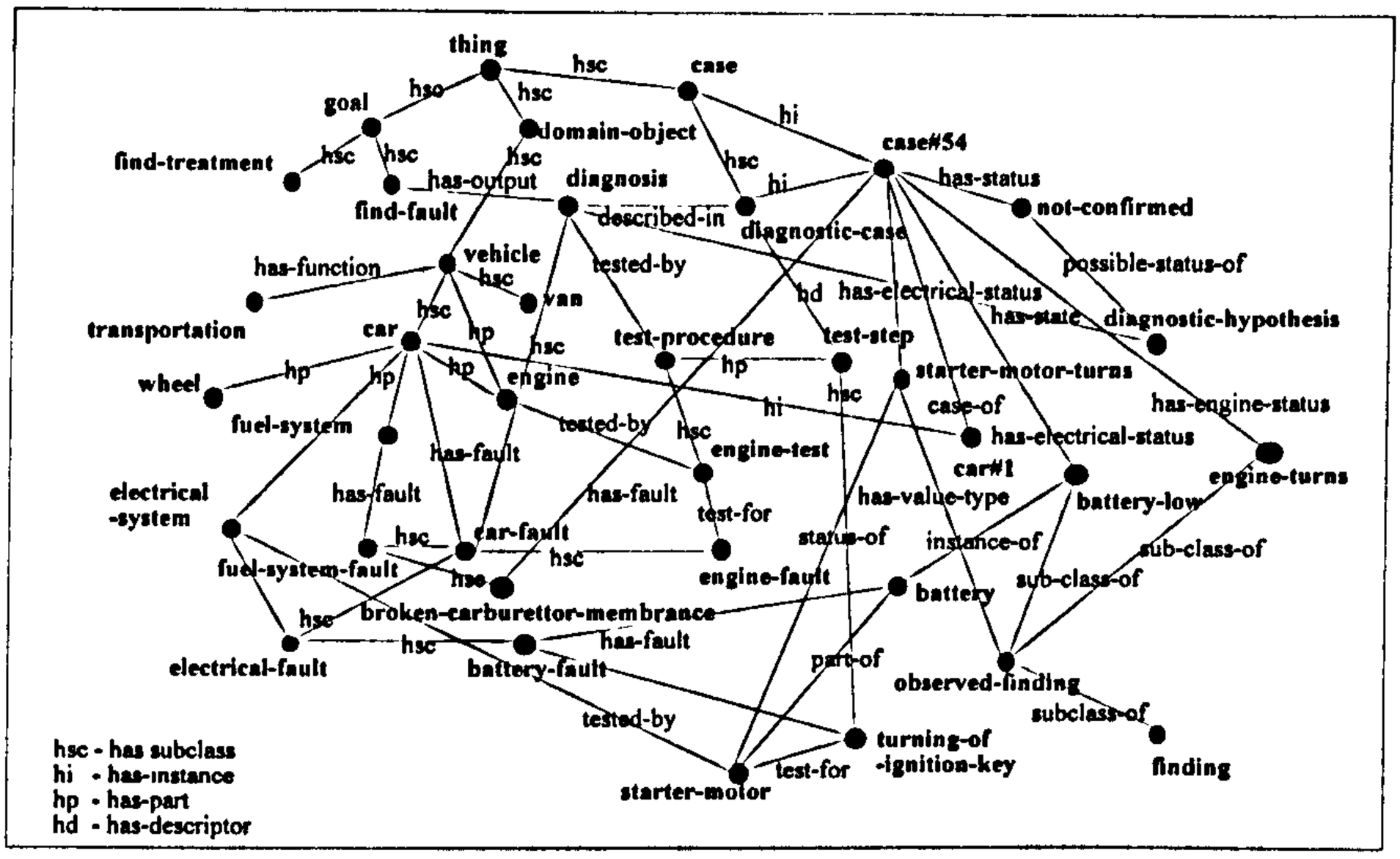


Figure 3.8 : A partial conceptual structure of the domain of car-starting problem represented by CREEK architecture
Source : Aamodt [1994a,1994b]

In PROTOS and CREEK architectures, the use of richer semantic relations for linking object concepts in order to form a conceptual knowledge basis for retrieving and selecting similar concrete experiences (cases) for biasing and narrowing the direction of problem-solving can provide a better solution for tackling problems in the open and weak theory domain[33]. However, these bases are only suitable for the diagnostic domains such as the hearing-disorder or car-starting fault diagnosing. This is because the features of a diagnosing symptom

---

[33] The problem of the open and weak theory domain does not necessarily imply that there is little general knowledge available, but that the general knowledge of that problem is theoretically uncertain, incomplete and subject to changes [Aamodt, 1994a]. For example, in PROTOS, a feature of the given symptom, such as *ac_reflex_u(normal)*, might be in two or more symptom categories, such as the *normal_ear* and *possible_menieres* categories [Bareiss, 1989: Figure 4.2: 60]; and in CREEK, a car-starting-fault might be caused by many factors such as broken-carburettor-membrane or carburettor-valve-stuck in fuel-system, the status of the distributor or low-battery in the electrical system, the weather conditions and the location of the car [Aamodt, 1994a].

and the situations of repairing a car are *concrete*, so that the relevant concepts in the bases can be activated by means of the pre-specified relations to retrieve the similar specific experiences and then to modify the selected case according to the concrete characteristics of the diagnosing case. Examples of the input diagnosing case are shown in Figure 3.9 for CREEK architecture and in Figure 3.10 for PROTOS architecture.

Case# 54

instance-of	value	car-starting-case diagnostic-case
has-task	value	find-car-starting-fault
has-status	value	not-confirmed
of-car	value	car#1
has-fault	value	broken-carburetor-membrance
has-fault-explanation	value	(.....)
has-repair	value	replace-carburetor-membrance
has-electrical-status	value	battery-low starter-motor-turns
has-engine-status	value	engin-turns engin-does-not-fire
has-ignition-status	value	spark-plugs-ok
has-weather-condition	value	low-temperature sunny low-air-presur
has-recent-driving-history	value	hard-driving

.....

Figure 3.9 : A partial representation of case#54
Source : Aamodt [1994a, 1994b]

Case : p8590R

Unknown

s_neural(mild,gt4k)	tymp(a)
s_neural(mild,lt1k)	speech(normal)
ac_reflex_u(normal)	air(mild)
ac_reflex_c(normal)	history(vomiting)
o_ac_reflex_u(normal)	history(dizziness)
o_ac_reflex_c(normal)	history(fluctuating)

Figure 3.10 : The features of a diagnosing symptom
Source : Bareiss [1989]

3.3.2 The General Process Cycle of Case-Based Reasoning

For reusing memories of previous similar cases (concrete experiences), two major processes are inherent in the CBR paradigm, as illustrated in Figure 3.11: the *recalling* and *modifying* processes. Consideration of the recalling process, which includes the finding and selecting sub-processes, involves retrieving the right case at the right time. Consideration of the modifying process, which includes the adapting and storing sub-processes, involves modifying the selected case and adjusting the case memory structure.

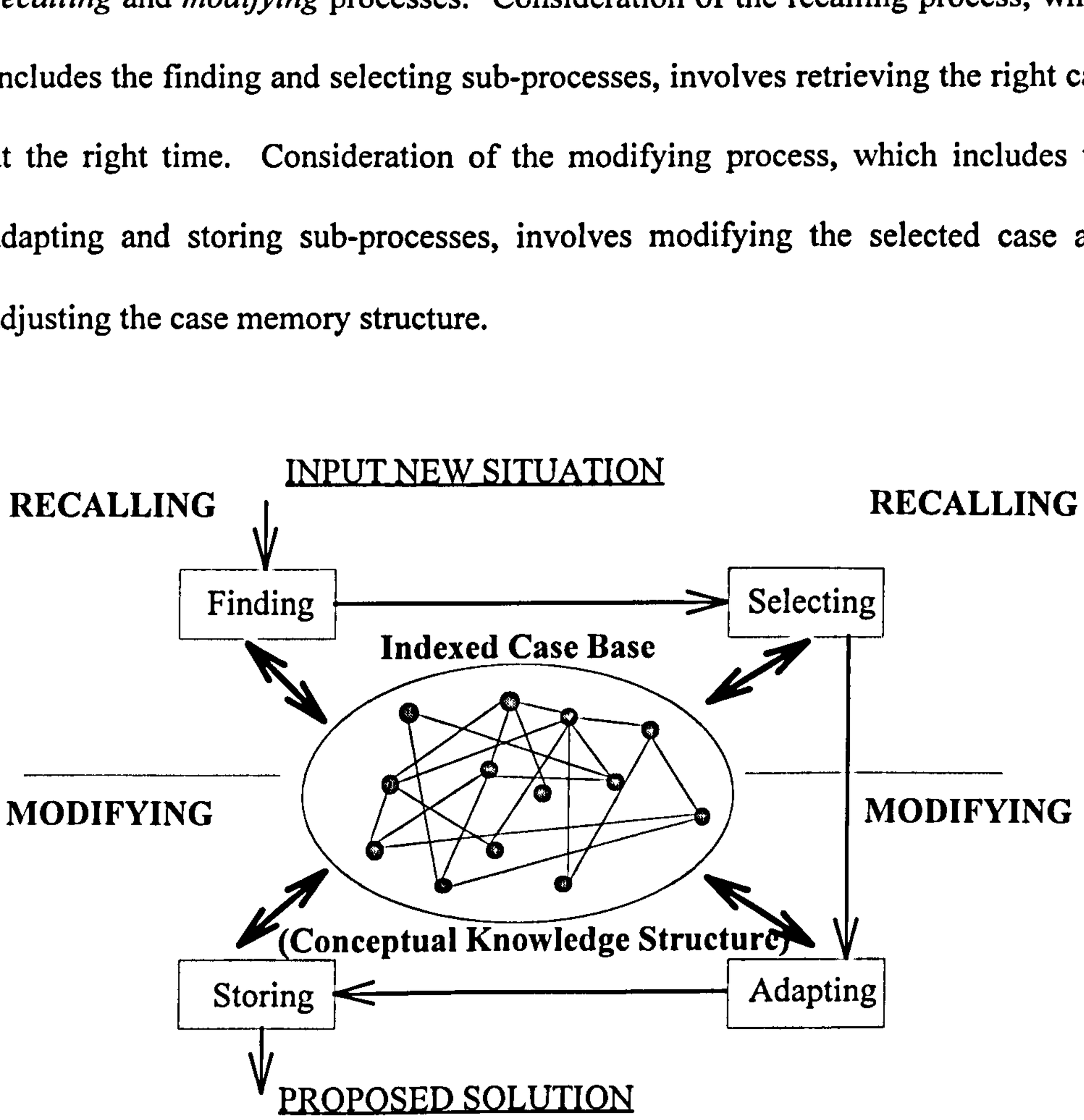


Figure 3.11 : The general process cycle of CBR

3.3.2.1 The recalling process

The recalling process involves accessing cases (reminding), which is the primary issue of CBR [Kolodner, 1996]. Consequently, the task of indexing cases is vitally important. The indices of a case are those features that are predictive[34] of something important in the case. This means that each case can be recalled by its own indices[35] whenever this is appropriate in the future. For example, in PROTOS, the reminding links can be used to identify the relevant symptom categories, and the difference links can be used to retrieve the other exemplars that consist of the unmatched features for the diagnosing symptom. In fact, indexing is very important not only for the recalling process but also for the modifying process, as will now be explained.

The case memory consists of a number of interrelated cases. Two sub-processes will be carried out in order to retrieve a case as a template for further refining. The *finding* of similar cases by means of the indices is the first sub-process. The criterion of whether a case is similar to the given problem depends on whether the matching strength is above a certain threshold[36]. For example, in PROTOS, the

---

[34] Whether a feature of a case is predictive is judged by the usefulness and importance of the feature in the applied domain, no matter whether it relates to surface, deep or structural features [Kolodner, 1996]. For example, the hair colour of an applicant for a modelling job is more important than it is for a computer programming job [Kolodner, 1984].

[35] The indices of a case need to be concrete enough to be recognisable and abstract enough to make a case useful in a variety of future situations [Kolodner, 1993].

[36] The value of the threshold is empirically set by the system builder. For example, in PROTOS, the threshold is set to 0.2 for the hearing-disorder domain [Bareiss, 1989].

normal_ear and *possible-menieres* symptom categories are the two possibilities after combining the recalling strength when the diagnosing symptom shown in Figure 3.10 is given. Two or more cases might be found which are similar to the given problem. The *selecting sub-process* is executed to evaluate the matching cases so that the best matching case will be returned as the outcome of the recalling process. The confirmation of the selection might be automated, as in PROTOS architecture, or interacted with the user, as in CREEK architecture.

Today, many CBR systems are dedicated to this process as problem-solving assistants that simply retrieve relevant prior concrete experiences for a user to consider while solving some problem. For example, CLAVIER, ARCHIE-II [Domeshek and Kolodner, 1993], MIDAS [Domeshek et al., 1994a], and CASECAD [Maher et al., 1995] are all retrieve-and-propose systems.

3.3.2.2 The modifying process

Indices themselves are not equal to the cases, but are more like "synapses"[37] of the cases. Using these indices, the recalling process returns the *best* matching case out of a set of finding cases, rather than the *exact* matching case. In the CBR paradigm, the returned partial matching case is viewed as a default answer that might be not quite right or incomplete. For finding a better solution to fit the current problem, the unmatched part of the selected case has to be settled or

---

[37] The term 'synapses' is a metaphor borrowed from neurophysiology.

modified by activating the *adapting sub-process*. For example, in PROTOS, the resolution of the unmatched part might be by means of the difference links for finding other exemplars including the unmatched part or the prototypicality links for choosing the next most appropriate exemplar, or even returning back to the next most suitable category for the diagnosing symptom; and in CREEK architecture, the modification of the faulty solution is carried out by activating the relevant concepts according to the findings given by the selected case to form possible explanations. Thus, these explanations are presented to let users confirm before looking for the next best matching case. Although, to date, many deliberate methods have been developed and adopted in the CBR systems, the strategies that control these methods for adaptation are hard to generalise, hard to implement, and easy to break[38] [Kolodner, 1993; Riesbeck, 1996]. Accordingly, the experts in CBR research and applications agree that the better solution in the applied CBR system is to create a shared environment between system and user so that the role of the user is to perform adaptation and evaluation [Kolodner, 1991, Barletta, 1994, Mark et al., 1996]. For example, in the recent CLAVIER [Hennessy and Hinckle, 1992] version, the modification of the unmatched part is by means of asking the operator to adapt manually rather than using the system to substitute the highest possible priority parts automatically from the list according to 'similarity'

---

[38] These phenomena are caused mainly by the encoded rules that are used to guide adaptation, indicating, for example, which aspects of a situation to adapt and how to control the adaptation process. For further details of the adaptation methods and strategies, see Kolodner [1993], Chapters 11-12.

criteria based on the part and its spatial context in the layout[39] [Barletta and Hennessy, 1989; Mark et al., 1996].

The CBR systems expand their knowledge by accumulating new cases and/or modifying parts of the structure of case memory. The proposed solution provided by the previous sub-processes will be compared to the new situation by the *storing sub-process* to determine whether a new case is constructed and/or the knowledge structure is refined. For example, PROTOS extends its knowledge in the following two ways. The first method is that of adding a new exemplar when it cannot classify the given diagnosing symptom or when the similar symptom category cannot be adequately explained. To integrate the new case into the conceptual knowledge structure, PROTOS acquires explanations and creates the indices by asking the teacher. The second method is that of refining the parts of the knowledge structure by adding new explanations for the unmatched features of the imperfect, but correct matched exemplar, or by increasing the prototypicality of the exemplar while the given diagnosing symptom is closely matched by the exemplar, etc. In CREEK architecture, while the experience from the problem just solved is retained as a new case in the case base, the explanation structures which contain the findings that can be used to explain the relevant findings from the problem-solving experience, and the solution itself are stored into the new case as indices for integrating into the conceptual knowledge fundament as well as the activated concepts as shown in Figure 3.12.

---

[39] The difficulty of the automated adaptation arises from the complexity of deciding thermodynamic compatibility from a part's spatial context [Mark et al., 1996].

case# 99 (New Retained Case)

```
instance-of              solved-case  car-starting-case
of-car                   N-VD-345699
...                      ...
has-relevant-finding     (has-electrical-status  electrical-system-ok
                          has-engine-status (engine-turns engine-does-not-fire)
                          has-ignition-status (spark-plugs-ok distributor-ok)
                          has-weather-condition (low-temperature moisty
                                                normal-air-pressure)
has-driving-history      ...
...
has-solution             (water-in-gas-tank
                         (0.92
                          ((engine-does-not-fire caused-by water-in-gas-mixture)
                           (water-in-gas-mixture caused-by water-in-gas-tank))
                           (carburettor-fault has-status false)
                           (((low-temperature combined-with moist) lead to
                                 condensation-in-gas-tank)
                            (condensation-in-gas-tank cause water-in-gas-tank))
                           (water-in-gas-tank does-not-violate
                              (electrical-system-ok spark-plugs-ok distributor-ok
                               engine-turns normal-air-pressure))
...
```

Figure 3.12 : A partial structure of a new retained case
Source : Aamodt [1994a]

Chapter 4

The Dual-Base Knowledge Model (DBKM)

The DBKM, based on the various theories of conceptual knowledge in the psychology realm, is proposed for handling the phenomena of the multiple-interpreting and polymorphic objects (relations between/among the same related objects) of reality, modelling the expertise of the experts in their memory, and emulating the behaviour of experts in the conceptual data modelling tasks. The proposed model includes two knowledge structures, namely the concept and case bases. The concept base is a vast and general web-like structure representing the real world as a total pattern in terms of the object nodes and relations. The case base includes a number of interrelated specific and concrete cases (finished schemata) that respectively represent a piece of reality (the UoD) in terms of entities and relationships. By means of the correspondence of these two structures, the concrete, finished schemata can be reused, and then the semantic interpretation can be fulfilled in an effective manner by virtue of this circumstance. The DBKM is a theoretical architecture which is used to break the limitations of the current KBDDSs in order to alleviate the identification and interpretation problems inherited from the intrinsic character of the conceptualisation. The theoretical view of the DBKM is presented in Figure 4.1.

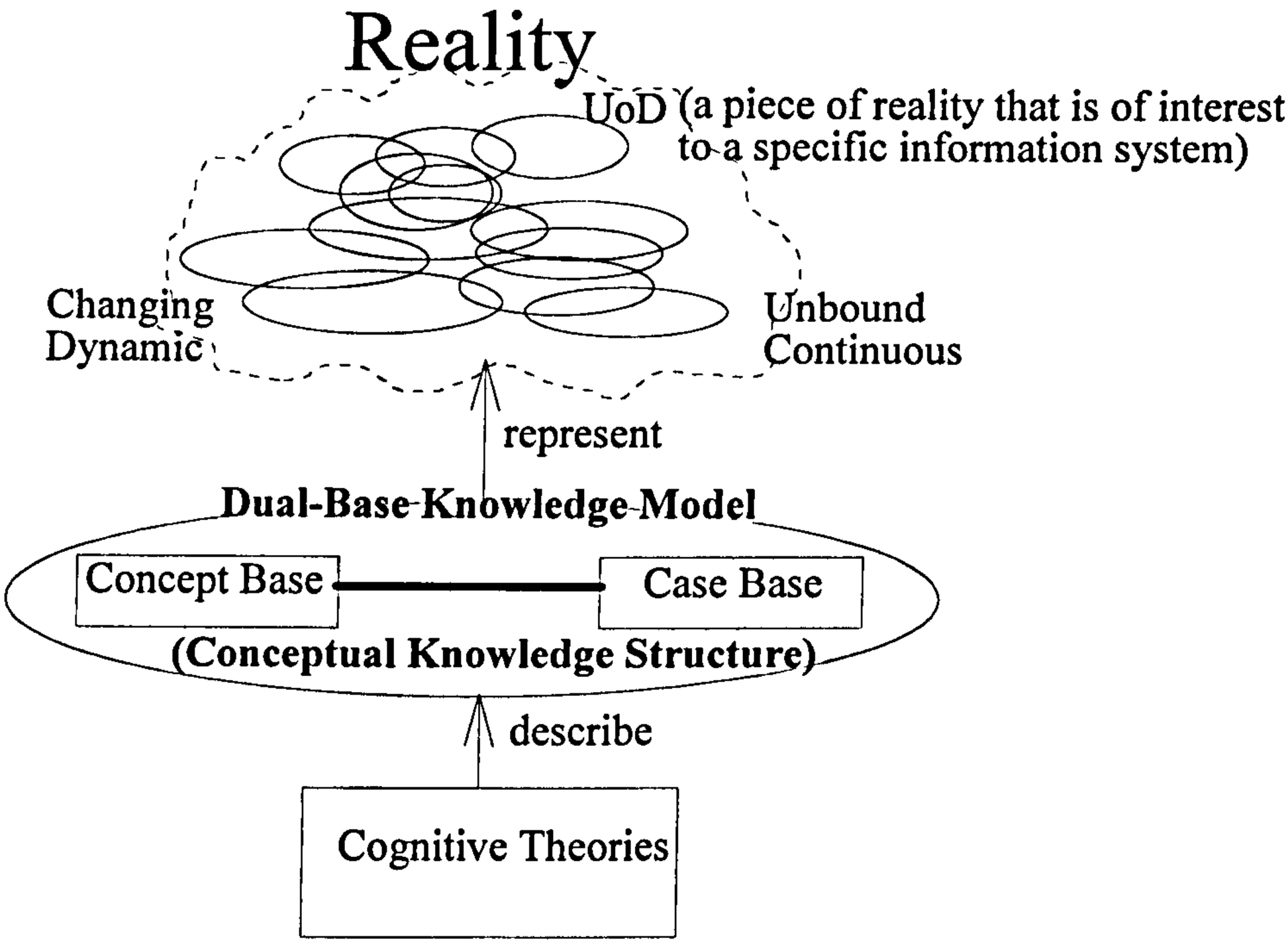


Figure 4.1 : A theoretical view of the dual-base knowledge model (DBKM)

The cognitive perspectives on the DBKM in terms of the cognitive representation, the cognitive process and the cognitive characteristics are illustrated in section 4.1. The structure of the DBKM, including the representations of the concept and case base, the intra-structure of the case base and the relevance of the concept and case bases, are presented in section 4.2. In section 4.3 the essence of the DBKM is characterised into four properties that can be used to facilitate the process cycle of the DBKM. The general representations of the two knowledge structures (the concept and case bases) and their relevance are formalised by set theory in section 4.4. Finally, the intrinsic properties of the relations in the DBKM for retrieving the relevant concrete experiences are discussed in section 4.4.

4.1 The Cognitive View of the DBKM

4.1.1 The Cognitive Representation of the DBKM

For database designers, the semantic model, as exemplified in the ER, EER models, is similar to the scene-structure mental model for psychologists, i.e. the semantic data model is regarded as a theoretical representation of a mental representation, representing the real world in terms of entities and relationships. A mental model is a theoretical representation of a mental representation that in turn represents an interesting aspect of the real world [Palmer, 1978]. Basically, from the psychological viewpoint a theoretical representation of the mental world focuses on how people *conceive* of the real world in some aspect rather than on how the real world is *actually* structured. Therefore, while studying the mental representations of conceptual knowledge, psychologists construct their own representations, such as three different views of concepts, schema, script etc., based upon certain cognitive phenomena. Although these constructions focus on different concerns, leading to multifarious views of how people mentally represent conceptual knowledge to categorise and understand the domain of objects or events, it can be asserted that 'people's mental representations undoubtedly include mental objects corresponding to real objects (or events) and mental relations that correspond to or capture the relations between real objects' [Cohen and Murphy, 1984: 31]. As shown in Figure 4.2, *mental representation*, including mental objects such as 1', 2', 3', etc. and mental relations such as

R'(1',2'), R'(1',4'), etc., has a correspondence to the *represented world* including real objects such as TV, sofa, wall, cabinet, picture etc. in the 'living room', or real events such as entering, ordering, eating, paying bill and leaving, etc. in the 'going to restaurant', and the relations such as picture [On] the wall, TV to the [Right] or [Left] of the cabinet, etc. in the 'living room', and ordering [Before] eating, eating [Before] or [After] paying, etc. in the 'going to restaurant'. Under a postulated, *theoretical representation* of mental representation, the mental objects and relations are embodied to represent the represented world in some aspect. For example, in the scene conceptual structure, the *mental objects* are represented as *object concepts*, and the *mental relations* are represented as the *spatial relations* among these object concepts; in the script conceptual structure, the *mental objects* are represented as *event concepts*, and the *mental relations* are represented as the *temporal relations* among these event concepts. Thus, the theoretical representation can be viewed as the embodiment of the mental representation that represents the real world in terms of the embodied mental objects and mental relations, i.e. the embodied mental objects correspond to the mental objects in the mental representation that in turn represent the objects (events) in the represented world; the embodied mental relations between/among the embodied mental objects correspond to the mental relations between/among the corresponding mental objects in the mental representation that in turn represent the relations between/among represented objects (events) in the represented world.

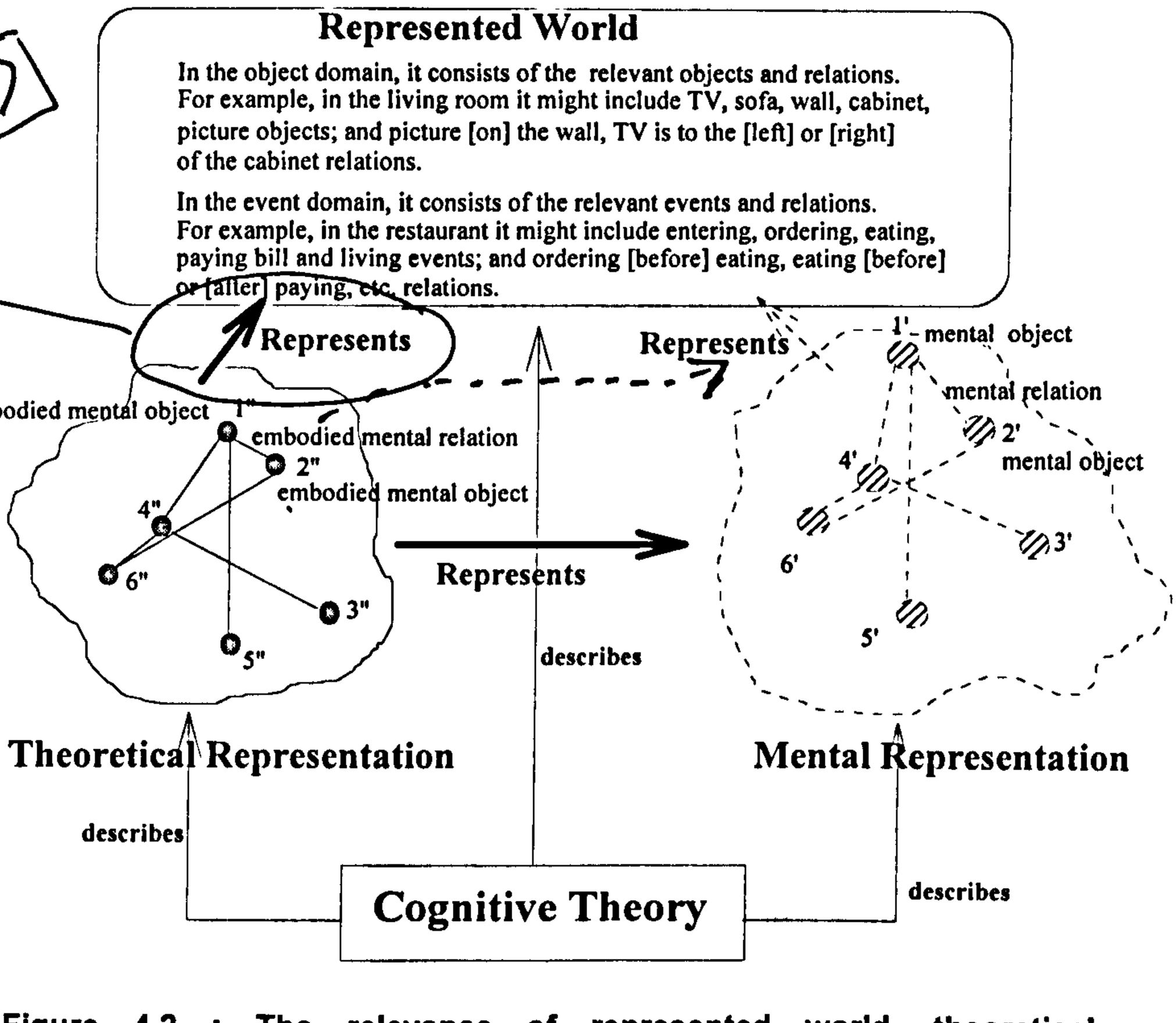


Figure 4.2 : The relevance of represented world, theoretical representation and mental representation

The reality, in terms of the objects and the relations among them, is multiplex and complex[40]. In this research, the notion of how to assist people to manage a chaotic reality involves regarding the mental representation as being crucial and of primary interest as the theoretical representation for handling the essence of the reality. That is, by means of linking these two representations, not only the significant features of the reality can be spotlighted but also the status of the reality can be stabilised.

---

[40] Because the subject of this research is focused on conceptual data modelling, the examples typified here are all from the view of the objects and relations between/among them in order to consider the essence of the reality. However, the analysis of the relevance of the two representations, mental and theoretical, illustrated in this section, is also applicable to the event situation.

The theoretical representation can be used as a template for modelling the represented world - the real world. For example, in the scene conceptual structure, it can be used to represent the living room scene of Mr. Johnson's and Mr. Brown's etc. In different living room scenes, although they are framed by the same structure, their contents might be different even if they contain the same real objects[41]. According to the correspondence between the theoretical representation and the mental representation, the embodied mental objects that all refer to the same real object, and the embodied mental relations between/among the embodied mental objects that refer to the same real object[42], can be handled by the corresponding mental objects and mental relations in the mental representation. For example, in Mr. Johnson's living room scene, the TV may be 19 inches in size, but it may be 28 inches in Mr. Brown' living room scene. The two TVs are all represented as a TV mental object in the mental representation. Two kinds of mental relations are included in the mental representation. First, there are undefined relations: for example, 'TV is [in front of] the sofa' in Mr. Johnson's living room and 'TV is to the [left] of the sofa' in Mr. Brown's living room are co-ordinated by an *undefined* link between TV and sofa mental objects[43] in the

---

[41] Whether two or more real objects are the same or different depends on the essence of those objects based on their purpose in the representation. For example, in the living room scene structure, the size and the type of the TV might be just attributes of the TV, and then no matter what the size or type of TVs, they are all regarded as a TV object. In the electricity shop scene, those features might be used to group a set of TVs. So, black and white TVs and colour TVs are different objects although they are all of the TV object.

[42] Of course, these embodied mental objects and relations are of different represented worlds.

[43] Although the name of the TV and sofa mental objects in the mental representation are the same as the TV and soft embodied mental objects in the different represented worlds - Mr. Johnson's and Mr. Brown's living room scenes - the contents of these two types of objects are different.

mental representation. Secondly, there are invariant relations: for example, the 'picture is [on] the wall' in Mr. Johnson's and Mr. Brown's living room is represented by the [on] invariant link between Picture and wall[44] mental objects in the the mental representation.

Because of the flexibility of the mental representation and the correspondence of mental and theoretical representations, it is possible to co-ordinate the multiplicity of the reality. Because of the normalising of the theoretical representation, it is possible to handle the complexity of the reality. In order to formulate these two representations, the structure of each representation and their relevance, as shown in Figure 4.3, may be illustrated as follows as a framework for constructing a knowledge model for the multiple interpretations and ploymorphic objects in the conceptualising task:

(1) The mental representation is a simple structure including the mental objects and relations among them. For example, in representing the living room domain (real world), the relevant mental objects such as the sofas, TV, fireplace, pictures, windows, etc. and the mental relations that may be invariant (for instance a picture is on the wall) or may be undefined (for instance a TV has a link to the sofa, irrespective of whether, in terms of relations, the TV is to the left or right of the sofa) are presented in the structure.

(2) The theoretical representation is an embodiment of the mental representation that is used to express the real world intentionally. The structure of this

---

[44] No matter which represented world the mental representation represents, the invariant relation between/among mental objects is always valid.

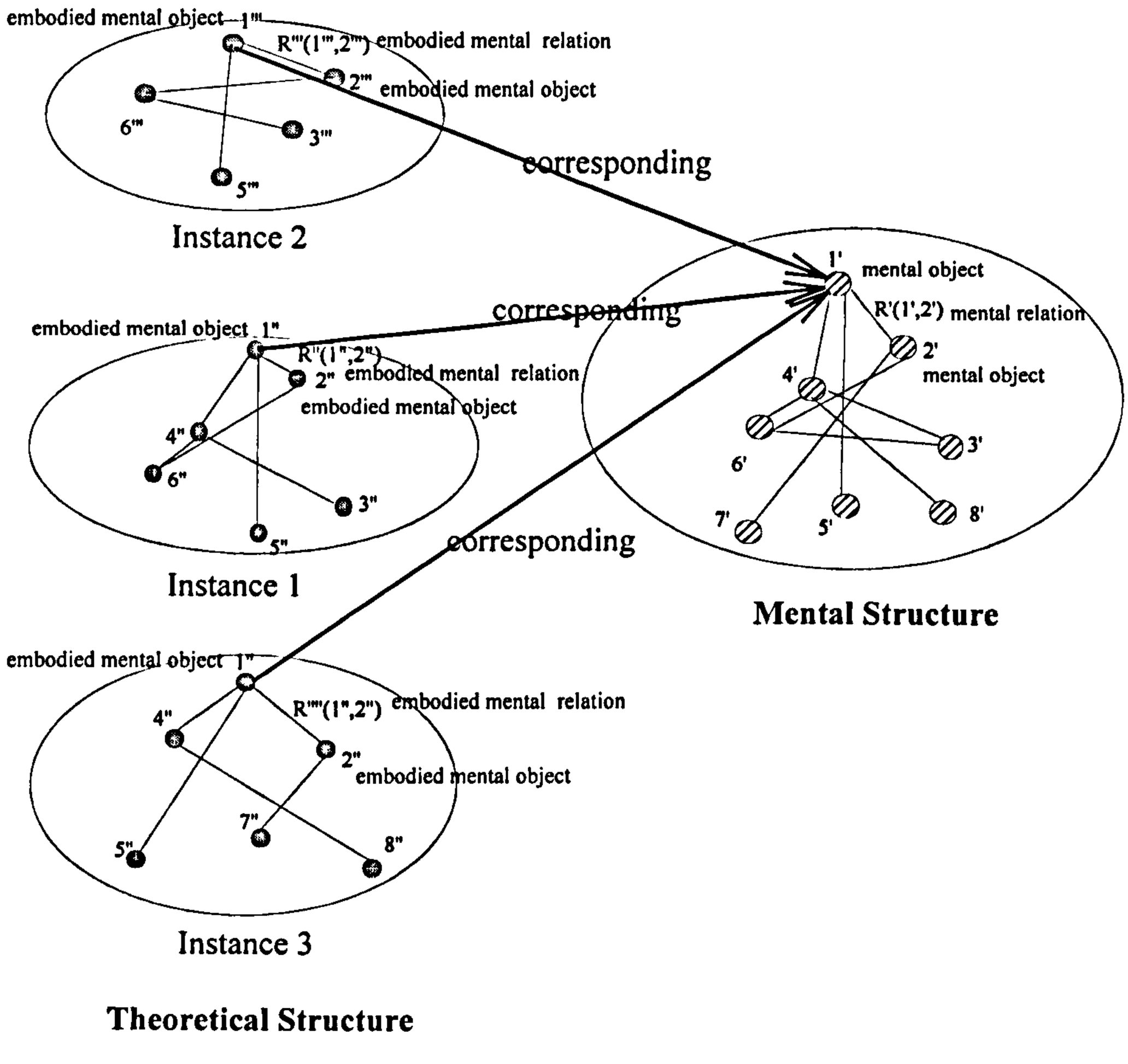
representation includes a number of instances consisting of the *embodied* mental objects and relations depending on what kinds of real world the representation wants to present. For example, in representing Mr. Johnson's living room, the embodied mental objects might be a TV (Sony), painting (Chinese), sofa (two-section), cabinet (book), china (three-colour), and the embodied mental relations might be the china is [on] the cabinet, the TV is to the [left] of the cabinet, the painting is [on] the wall that is [above] the sofa, etc.

(3) There exists a mapping function (correspondence) from the *embodied* mental objects and mental relations in the theoretical structure to the *mental* objects and relations in the mental structure that in turn represent the *real* objects and relations in the real world. For example, the embodied mental objects 1' in instance 1, 1''' in instance 2, and 1'' in instance 3, correspond to the mental object 1' in the mental structure that in turn represents the represented real object 1. The embodied mental relations R''(1'',2'') in instance 1, R'''(1''',2''') in instance 2, and R''''(1'',2'') in instance 3, correspond to the mental relation R'(1',2') in the mental structure that in turn represents the relation of real object (1) and (2) in each represented world.

overloaded

Must be an instance of the total structure? OK see 4.3

1' in the mental structure



A. The mental structure consists of mental objects such as 1', 2', 3', 4', 5', 6', 7', 8' and mental relations such as R(1',4'), R(1',2'), R(1',5'), R(2',6'), R(2',7'), R(3',4'), R(3',6'), R(4',6'), R(4',8') that might be invariant and undefined.

B. The theoretical structure includes three instances consisting of a part of the embodied mental objects and relations.
For example, instance 1 consists of the embodied mental objects 1'', 2'', 3'', 4'', 5'', 6'' and embodied mental relations R''(1'',2''), R''(1'',4''), R''(1'',5''), R''(2'',6''), R''(3'',4''), R''(4'',6'').

C. The embodied mental objects1'' in instances 1, 3 and 1''' in instance 2, all correspond to the same mental object 1', in the mental structure and the embodied mental relations R''(1'',2'') in instance 1, R'''(1''',2''') in instance 2 and R''''(1'',2'') in instance 3, that may be the same or different all correspond to the mental relation R'(1',2') in the mental structure.

D. A mental object and its corresponding embodied mental objects all represent the same real object or event; a mental relation between (among) mental objects and its corresponding embodied mental relations between (among) the corresponding embodied mental objects represent the relations between the represented objects (events) in their own right.

Figure 4.3 : The structures of the mental and theoretical representations

4.1.2 The Cognitive Process of the DBKM

Conceptual data modelling is known as a cognitive process. Conceptual data modelling is a process of building an adequate representation of a problem using particular modelling constructs. The cognitive behaviour behind this construction process involves incorporating the external part of the problem representation that is built as a semantic data model, and the internal part of the problem representation that refers to the appropriate experiences in our long-term memory[45] [Batra and Davis, 1992; Srinivasan and Te'eni, 1995]. That is, by using the previous knowledge stored in long-term memory to comprehend the problem and to elicit other relevant information from external knowledge sources, the represented world (a specific UoD) is modelled in terms of the entities (attributes) and relationships between (among) entities by the formal primitives supported by a particular semantic model. Thus, during the modelling, a set of cognitive activities, such as seeking and translating the information from external sources in the light of the previous knowledge that is internally represented in the human memory, using a semantic data model to externally represent the entities (attributes) and relationships of the specific UoD, and refining the internal and external part of the problem representation, are interactively executed until a satisfactory solution is reached, i.e. a finished schema is a product of the cognitive process in the human mind.

---

[45] The term 'parts' used here does not imply mutually exclusive parts [Srinivasan and Te'eni, 1995].

With reference to the long-term memory's capacity to store a number of previous experiences as a kind of black-box, Batra and Davis [1992] explored the behaviour of experts and novices in a conceptual data modelling task and found that the experts relied upon the referent experiences to help them generate a holistic comprehension picture of a problem before developing the conceptual model. The experts also had a richer and more differentiated vocabulary of facets to help them recognise certain requirements.

This research further extends the findings of Batra and Davis [1992] by adopting other researchers' notions of how expertise is represented in long-term memory and how this type of knowledge is retrieved from memory. Chase and Simon [1973], Dreyfus and Dreyfus [1986], and Lesgold et al. [1988] exemplified some cases to show that the expertise in experts' long-term memory is represented as many more unanalyzable concrete experiences. For example, chess masters can recognise about 50,000 different configurations of chess [Chase and Simon, 1973, Dreyfus and Dreyfus, 1986], and X-ray diagnosis experts have experienced about 200,000 X-ray pictures that display the different patterns of dark and light regions connected to particular symptoms before they can interpret the films intuitively [Dreyfus and Dreyfus, 1986; Lesgold et al., 1988]. In conceptual data modelling, every relevant object in a UoD is interpreted as an entity constant, and every relation between/among the relevant objects in a UoD is interpreted as a relationship constant. The finished conceptual data schema is a model of a specific part of reality. Thus, each schema represented by the semantic model is an experience about a piece of reality. That is, a finished schema representing the

↑ at instance level !

specific UoD is viewed as an instance of the particular representing structure (semantic data model).

Thus, while making a decision, experts just trigger the similar whole patterns for comprehending and solving problems at hand [Dreyfus and Dreyfus, 1986; Patula, 1992]. A similar whole pattern (finished schemata) is regarded as a plausible answer to the problem, but because of the possibility of new requirements, the answer might be incomplete or not quite right. The experts use this visible similar pattern to retrieve further relevant patterns in their memory or to elicit other information from an external knowledge source. Thus, it is reasonable to think that there is another structure as a source to enable the experts to obtain these patterns (knowledge) or information easily. In this research, the position of mental structure performs this role. According to the characteristics of the theoretical structure which correspond to those of the mental structure, experts use the mental objects corresponding to the entities in the referred concrete experience to retrieve other relevant patterns, including the new requirements. To retrieve some relevant chunks in the long-term memory, the notion of a semantic-based mental structure used here is consistent with, but more elaborated than, the hypothesis of 'conceptual chunking'[46] [Egan and Schwartz, 1979] and the notion

---

[46] 'Conceptual chunking' is like the 'P-Schema (scene)' that uses the spatial relations to integrate the relevant conceptual categories as a whole unit. For example, power supply chunking might include a source, rectifier, filter, regulator, etc., general concepts, and the relations among these conceptual categories (depend on the electrical theory). In the symbolic circuit drawing domain, the skilled electronics technicians use 'conceptual chunking' to construct the missing items (electrical logical symbols such as transistor, resistor, capacitor, etc.) in the answer sheets by recalling the functional units, such as amplifier, power supply, filter, rectifier, etc. in the long-term

of the implicit semantics of the mental categories[47] [Chase and Simon, 1972;1973].

The idea of facilitating the object identification and interpretation of conceptual data modelling by means of modelling the expertise in the experts' memory and emulating the behaviour of experts in the conceptual data modelling work, may be illustrated as follows :

(1) The semantic data model - a common framework, such as the ER, EER models - is regarded as a theoretical representation that represents the real world in terms of entities and relationships.

(2) The structures of mental and theoretical representation are further refined in terms of the following two structures respectively: the semantic-based mental structure, which is a refined mental structure, and the experience-based theoretical structure, which is a refined theoretical structure.

(3) Each finished schema is regarded as a whole pattern in this structure. The experience-based theoretical structure stores a number of finished schemata (concrete experiences) represented by a particular semantic data model.

---

memory. In other words, while seeing the answer sheets, the skilled electronics technicians use the rather sketchy surface features (the electrical logical symbols) in the sheet as a clue to construct the drawing by means of the 'conceptual chunk' which recalls the relevant functional units that might include the missing items.

[47] Although they do not further illustrate what the structure of the semantics of mental categories associated with the chunks is, they provide a general view of the way in which the intact and well-organised chess configurations or patterns stored in long-term memory are recalled to assist decision-making. In other words, while seeing the chess positions on the board, the chess master uses the surface features (the positions) of the board as a clue to decide the next move by means of the associated semantics of mental category to recall similar chunks.

(4) The semantic-based mental structure, consisting of general concepts and some types of semantic relations, is used to unify the number of concrete stored experiences in order that the similar experiences can be recalled at the right time

(5) The recalling process is from the semantic-based structure that includes general concepts and semantic relations, to the experience-based structure that consists of a number of concrete experiences.

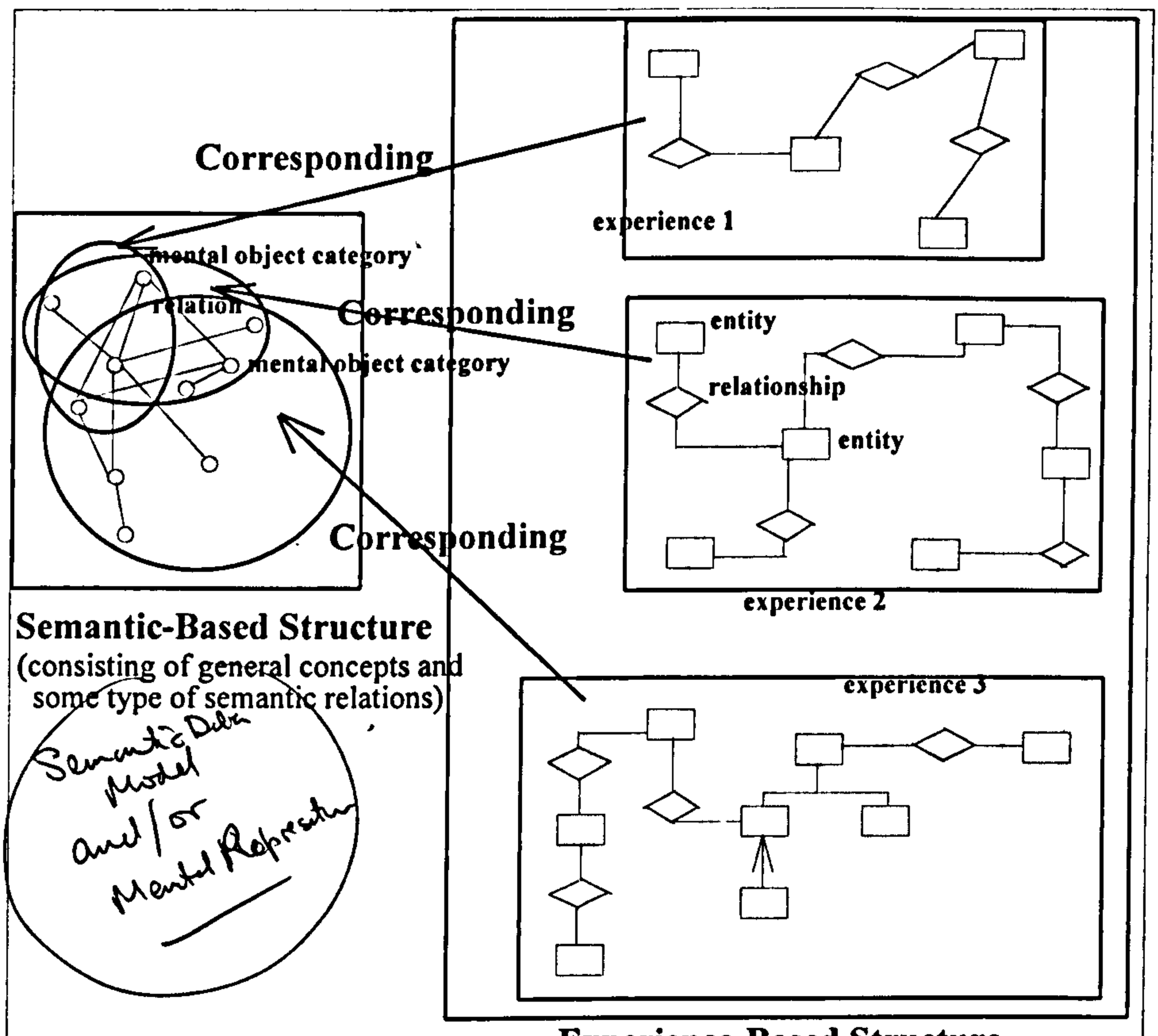


Figure 4.4 : The refinement of the mental and theoretical structures: the semantic-based mental structure and experience-based theoretical structure

4.1.3 The Cognitive Characteristics of the DBKM

In conceptual data modelling, the standpoint of the database designers about object concepts is shifting from *whether* an entity represented by attributes and relationships is an example (instance) of a particular object concept and *how* people decide whether an entity is of a particular object concept to *how designers decide whether the well-defined content and roles of an entity that is of a particular object concept satisfy their intended purpose*. During the conceptual data modelling process, the role of the database designers is not that of detached observers of the represented world, but is tightly coupled to the represented world. The meanings of concepts are not inherent in the represented world, but depend in part on some interpretation by the designers. In other words, the entities and relationships which designers discover in the represented world are not a matter of observation, but of interpretation[48]. Thus, the contents and roles of object concepts are heavily influenced by how people perceive the represented world, i.e. the meanings of object concepts are defined by *people's knowledge* about the world [Murphy and Medin, 1985], and the contents of object concepts are determined by their functional role that must be perceived *nonsolipsistically*[49], depending on the circumstances in which they are conceived in people's minds [Harman, 1987]. Consequently, an object category (thing in the real world) can be

---

[48] Of course, this does not mean subjectivism because the interpretation is based on the culture and the conventions of the represented world.

[49] The term 'nonsolipsistically' means that the contents of concepts are decided by their involved relations to things in the external world rather than being considered objectively fixed and unchangeable without reference to things in the external world.

interpreted in many ways according to its purposes in the represented worlds, and then the corresponding object concept (mental category) can also be embodied in many forms that respectively represent the represented object category in their own right. That is, object concepts in conceptual data modelling are inherently polymorphic - their corresponding entities vary in interpretative meanings (contents and roles) (see the discussion in section 2.1.1). Harman [1987] differentiated object concepts into two kinds, general and individual concepts, to explain why people can handle objects as things of an appropriate sort in certain circumstances. An individual concept functions in a certain environment to spot a particular object in the represented world to which a general concept applies. In this research, the individual concepts are viewed as the exemplars of their corresponding general concepts in the specific reality[50], i.e. the meaning of a general concept is represented by its *specific exemplars* extensionally [Brooks, 1978; Medin and Schaffer, 1978].

According to above discussion of the essence of the reality and the problem-solving behaviour of experts in the conceptualising task from the cognitive perspectives, the groundwork of this research, that is, investigating a knowledge model in order that the finished conceptual schemata or part of them can be reused and so help the difficulties of the conceptual data modelling work, may be characterised as follows:

---

[50] In other words, the UoD is a basic unit. Thus, the people in the specified UoD are regarded as individuals in contrast to other people in other UoDs. The meaning of a concept is bounded by the conventions and culture that are shared by the individuals in the specific environment or organisation.

(1) The DBKM includes two non-equivalent representation structures [Palmer, 1978]. One is the concept base, a semantic-based mental structure, that models the mental world, namely S_c, and the other is the case base, an experience-based theoretical structure, namely S_e, that consists of a number of narrow cases (finished schemata) representing the different views of reality.

(2) The S_c is a vast and general web-like structure, formed by object-type concepts and relations, that just displays the names of objects and indicates whether there are connections between (among) objects. The roles of this base are threefold: (1) to express the general meanings of objects and relations; (2) to co-ordinate the variety of objects; and (3) to provide a guide to recall by means of the relatedness of objects.

(3) The S_e includes a number of interrelated narrow cases representing the specific UoD, i.e. $S_e = \{s_{ei} \mid i \in N\}$, where s_{ei} is a whole pattern of a specific reality represented by a particular semantic model as a concrete experience. The entities in the S_e are the exemplars of the corresponding object-concepts in the S_c.

(4) A s_{ei} simultaneously represents the specific reality (UoD) and part of the mental structure (the concept base) in terms of the entities and relationships. In other words, an *entity* is an *embodied object* from the viewpoint of the concept base and an *interpreted object* from the viewpoint of the reality; a *relationship* between/among the entities is an *embodied relation* between/among the corresponding embodied objects from the viewpoint of the concept base and an

interpreted relation between/among the corresponding interpreted objects from the viewpoint of the reality.

(5) While designing a new schema, the designers recall a similar finished schema to get a whole picture of the problem-solving task, and then they use this whole pattern for comprehending the problem and for retrieving other relevant patterns or eliciting further information from external knowledge sources to model the represented world (UoD).

(6) The conceptual data modelling task in the DBKM is viewed as the *embodiment* from the general, vague object concepts (relations) in the concept base to the specific, concrete entities (relationships) in the case base, i.e. the conceptual schema is the product of embodiment by means of interaction between the concept and case base.

Bit confusing. Here since we are working with representations of these things!

There is a flip between mental mod and its representation a the conceptual schema that is an entity/rel structure.

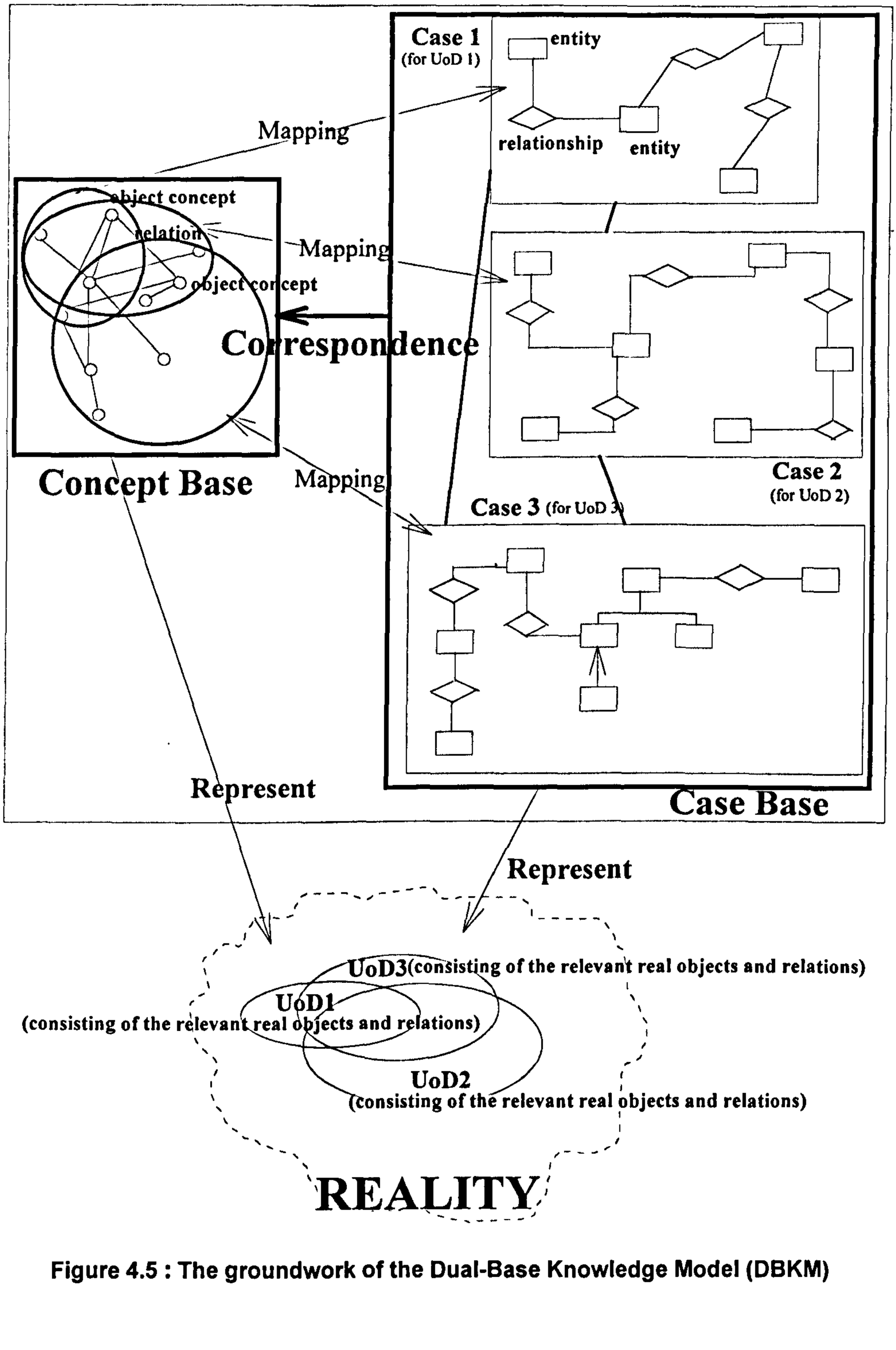


Figure 4.5 : The groundwork of the Dual-Base Knowledge Model (DBKM)

4.2 The Structure of the DBKM

Two issues are involved in experience reuse: these are *design by reuse* and *design for reuse*. How to *recall* the useful experiences from the DBKM and how to *retain* the newly gained knowledge into the DBKM are the main concerns of the former that will be discussed in the next chapter (Chapter 5); and the latter issue is concerned with how to *structure* knowledge, including representation, segmentation and indexing, and is discussed below.

In this research, the conceptual knowledge fundament of how to reuse the concrete experiences (finished schemata) to ease the confusion of object identification and interpretation is modelled as two separate but corresponding representing worlds: the concept and case bases.

4.2.1 The Concept Base

In this research, *concept* is regarded as a mental representation of a simple class such as book, student, etc. or a complex class such as transportation centre, control system, etc. [Rey, 1983; Medin and Smith, 1984] and is the atomic unit in the formation of the static part of general domain knowledge.

The concept base is viewed as a semantic-base mental structure. It is a *web-like* structure constructed by nodes and links. Each node in the network corresponds to

a concept and each link represents a relation between nodes. Each node is labelled with its corresponding concept name, and two types of links, namely the *is-a* and *unlabeled* links, are available for describing the relations between nodes in the concept network. An example of the concept base for a university domain is given in Figure 4.6.

(1) Node

using the concept name directly, such as the (Book) node representing the BOOK general concept.

(2) Links

1.*is-a* link: representing the relation among concepts in the taxonomy of concepts [Rosch et al., 1976b]. For instance, Teacher is a [*is-a*] Person. Professor, Reader and lecturer are all [*is-a*] Teacher. Within this taxonomy, Teacher is a basic object, Person is a superordinate concept, and (Professor, Reader and lecturer) are subordinate concepts. In other words, the [*is-a*] links represent the *generalisation* relations between concepts. For instance, the concept PUBLICATION is a generalisation of the concept BOOK, since every book is a publication. Likewise, the concept PERSON is a generalisation of the concepts STAFF, TEACHER, and STUDENT.

2.*unlabled* link: representing an undefined relation between concepts. Unlabelled links give the flexibility of relations between concepts according to the needs in a specific UoD. For instance, the concept BOOK and concept AUTHOR have an unlabeled link between them. This unlabeled link only says that there is a relation between them. In one context (e.g. the library domain) the relation between these two concepts might be [WRITTEN-BY]

relationships and the cardinalities may be as follows: BOOK (entity) to [WRITTEN-BY] (relationship) is of (1,n) and AUTHOR (entity) to [WRITTEN-BY] (relationship) is of (0,n). In another context (e.g. the bookstore domain) the relation between these concepts could be [EDITED-BY] (relationship) and the cardinalities could be: BOOK (entity) to [EDITED-BY] (relationship) is of (1,n) and AUTHOR (entity) to [EDITED-BY] (relationship) is of (1,n).

The concept base as a whole represents the general domain knowledge. It is used for thinking and reasoning and co-operates with cases in the case base to support the object identification and interpretation subphase of the conceptual data modelling task.

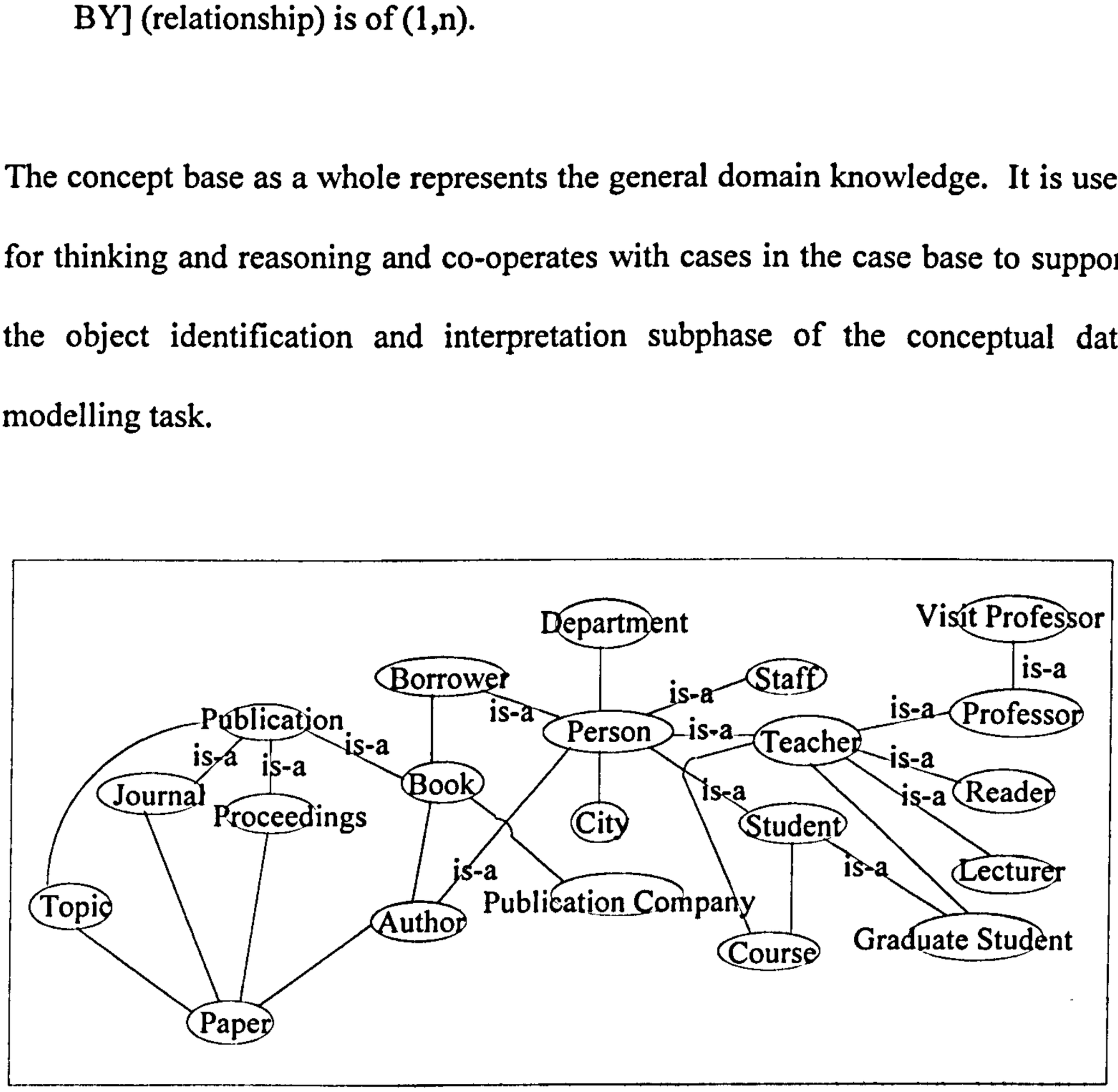
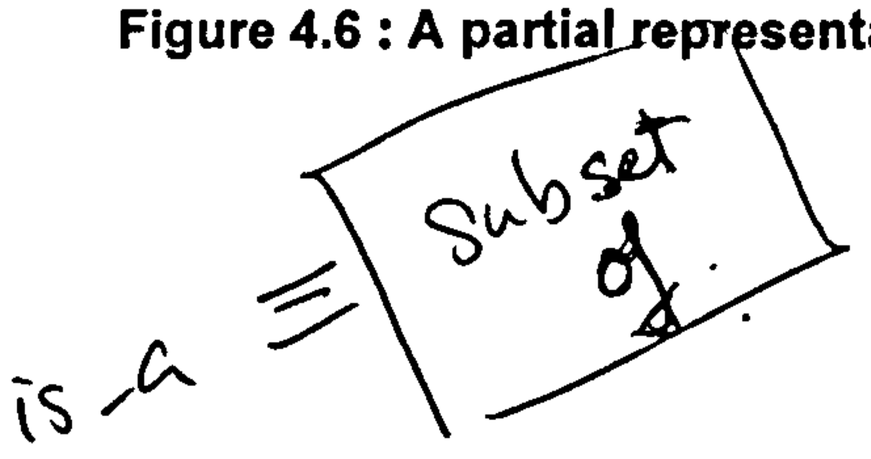


Figure 4.6 : A partial representation of the concept base for the university domain



4.2.2 The Case Base

4.2.2.1 The general contents and representation of a case

In this research, *case* refers to a previously experienced situation which has been captured and learned in such a way that it can be reused in solving future problems [Malek and Labbi, 1995]. In general, there are three major parts for any case: the problem description, the solution, and the resulting state of the world when the solution is carried out [Kolodner, 1993]. It is also true that the structure of a case may vary from one application to another. By means of the retrieval of old but correct schemata (syntactic and semantic) in the same system or analogous domains we can seek to resolve or interpret a new problem. The contents of a case are divided into three main parts: (1) the header (domain, subdomain, system name and its corresponding part of the concept base); (2) the main body (entities, relationship, and attributes); and (3) the links (proper subset-superset and overlapped link).

Ordinarily, the main body of a case described in the EER model can be written as a set of entities and relationships defined as follows: case = {<entities>, <relationships> }. The purpose of this research is to reuse the conceptual schemata by means of the concept base that can co-ordinate the multiple-interpreting objects and relations of the real world. Thus, from the viewpoint of the concept base, the main body of a case can be represented by the concepts and

the relevant relations as follows: case = {<concepts>,<relations>}. Therefore, the case containing three entities and two relationships, as shown in Figure 4.7, can be written, in general, as follows from the viewpoint of the concept base.

Case = {A, B, C, R(A,B), R(B,C)}
A, B, C : concept names
R(A,B) : relation between concepts A, B
R(B,C) : relation between concepts B, C

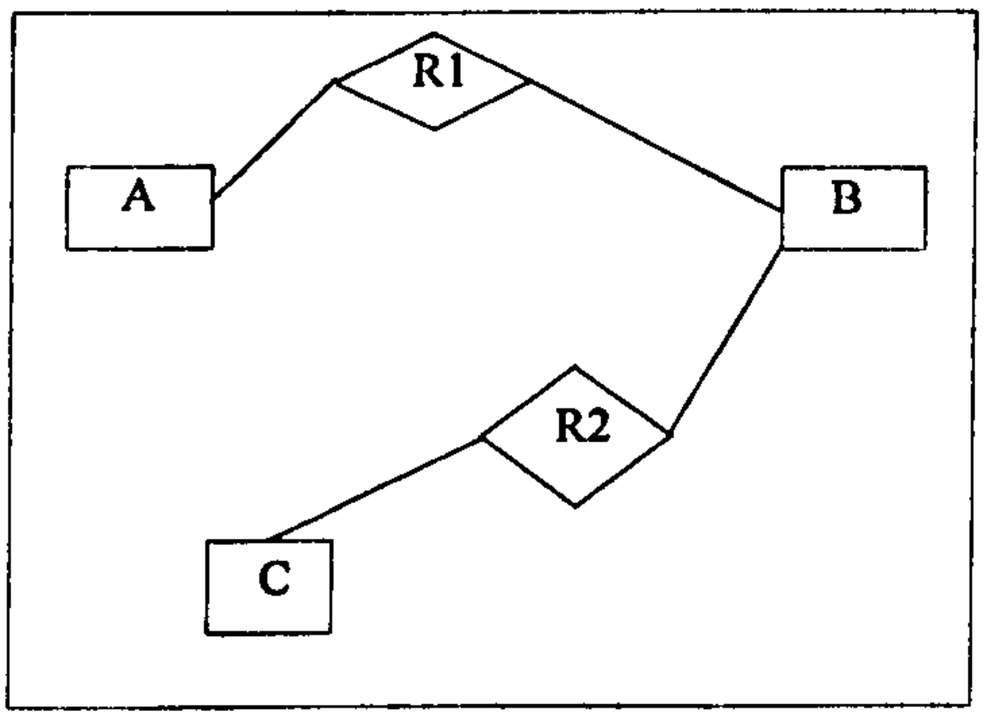


Figure 4.7 : The example of the ER diagram

4.2.2.2 The structure of the case base

Two types of case are organised into the case base in different ways: namely *domain* and *system* cases. Domain cases serve the purpose of analogous reference, and the system cases are for similar reasoning. An essential difference between these two types of cases is that the entities of domain cases are all in the superordinate level, i.e. their correspondences are all superordinate concepts in the concept base, but the entities of system cases are in all levels - the superordinate, basic and subordinate, i.e. their correspondences are spread over the taxonomy of concepts.

The relationship between a domain case and system cases

A domain case is generalised from two or more system cases in the same domain, which may correspond to *macrostructure* in the sense suggested by Kintsch and van Dijk [1978], i.e. each of these system cases is assumed to be an instance of the general domain case. Gick and Holyoak [1983] showed that using this abstract structure in addition to concrete examples might facilitate the analogical thinking. An example of these two types of cases and the corresponding concepts of their entities is shown in Figure 4.8: a transportation control domain case is abstracted from two different system cases - the train station and airport control system cases.

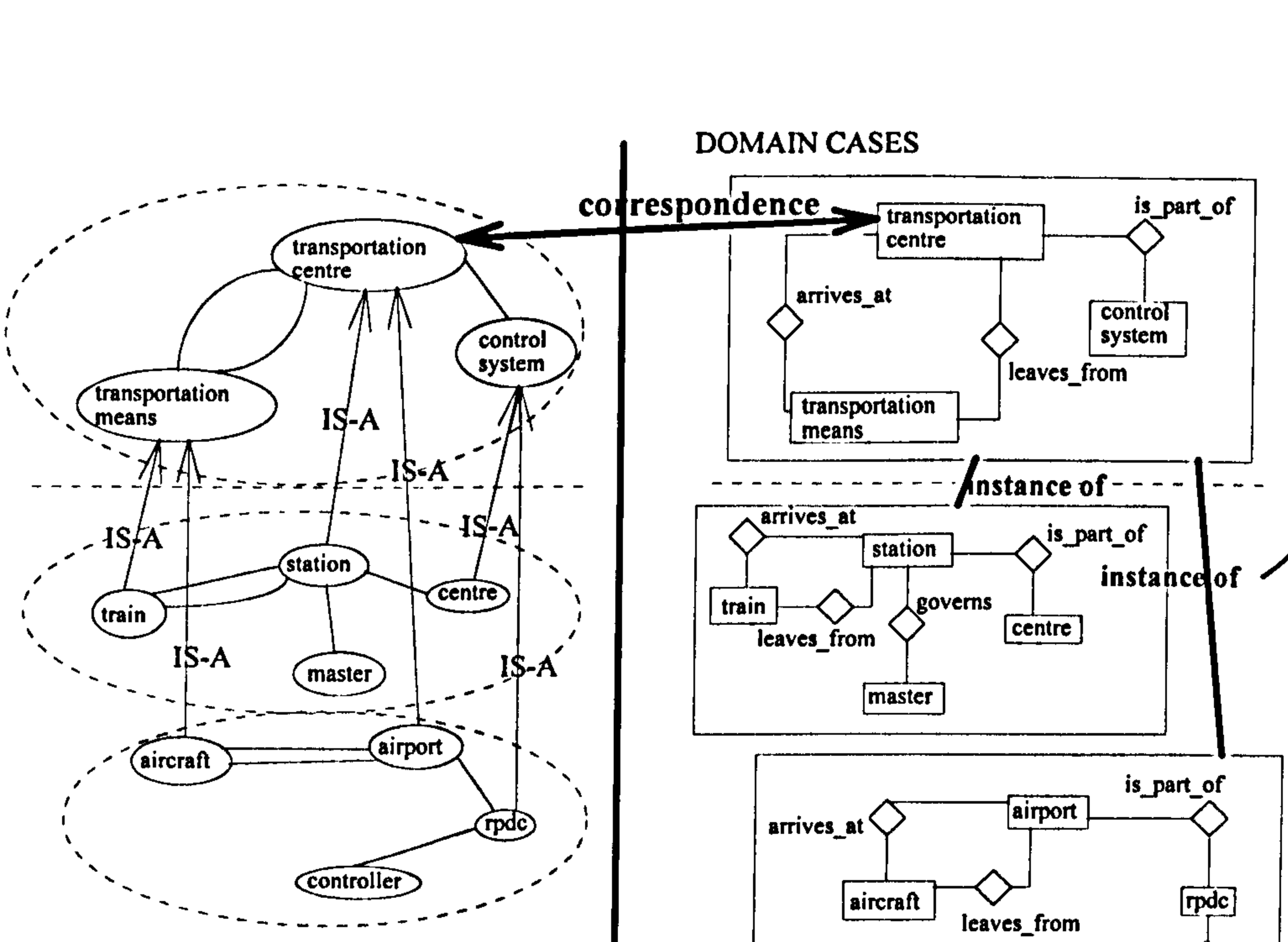


Figure 4.8 : A view of the two types of cases and the corresponding concepts of entities (for simplification only the correspondence of the transportation centre is shown)

The relationship between system cases

From the viewpoint of the concept base, there are two kinds of relations among the system cases in the case base: overlapped and proper subset-superset relations (the mathematical terms), as shown in Figure 4.9. These relations can be of help in reusing the old cases and facilitating the acquisition of new cases (experiences).

In the following, we use a set theory to define the relations among system cases:- (The more precise definition will be formalised in the section of the fomalisation of the dual-base knowledge structure)

Definition: Proper subset-superset relation (abbreviated as **sub-super** relation)

X is a set of cases. Let R_1 be a sub-super relation on X and A, B $\in$ X. The relation can be defined as follows.

This needs to be defined ok

$$A \; R_1 \; B \Leftrightarrow A \subset B$$

If A R_1 B, then A is called the proper subset of B and B is a super set of A. The properties of this relation are as follows:

1. Irreflexivity : $\forall A \in X,\ A \; R_1 \; A$
2. Asymmetry : $\forall A, B \in X,\ A \; R_1 \; B \Rightarrow B \; R_1 \; A$
3. Transitivity : $\forall A, B, C \in X,\ (A \; R_1 \; B)$ and $(B \; R_1 \; C) \Rightarrow A \; R_1 \; C$

Definition: Overlapped relation (abbreviated as **overlp** relation)

X is a set of cases. Let R_2 be an overlp relation on X and A, B $\in$ X. The relation can be defined as follows.

$$A\ R_2\ B \Leftrightarrow A \cap B \neq \Phi \text{ and } A\ R_1\ B,\ B\ R_1 A.$$

The properties of this relation are:

1. Reflexivity : $\forall A \in X,\ A\ R_2\ A$
2. Symmetry : $\forall A, B \in X,\ A\ R_2\ B \Rightarrow B\ R_2\ A$

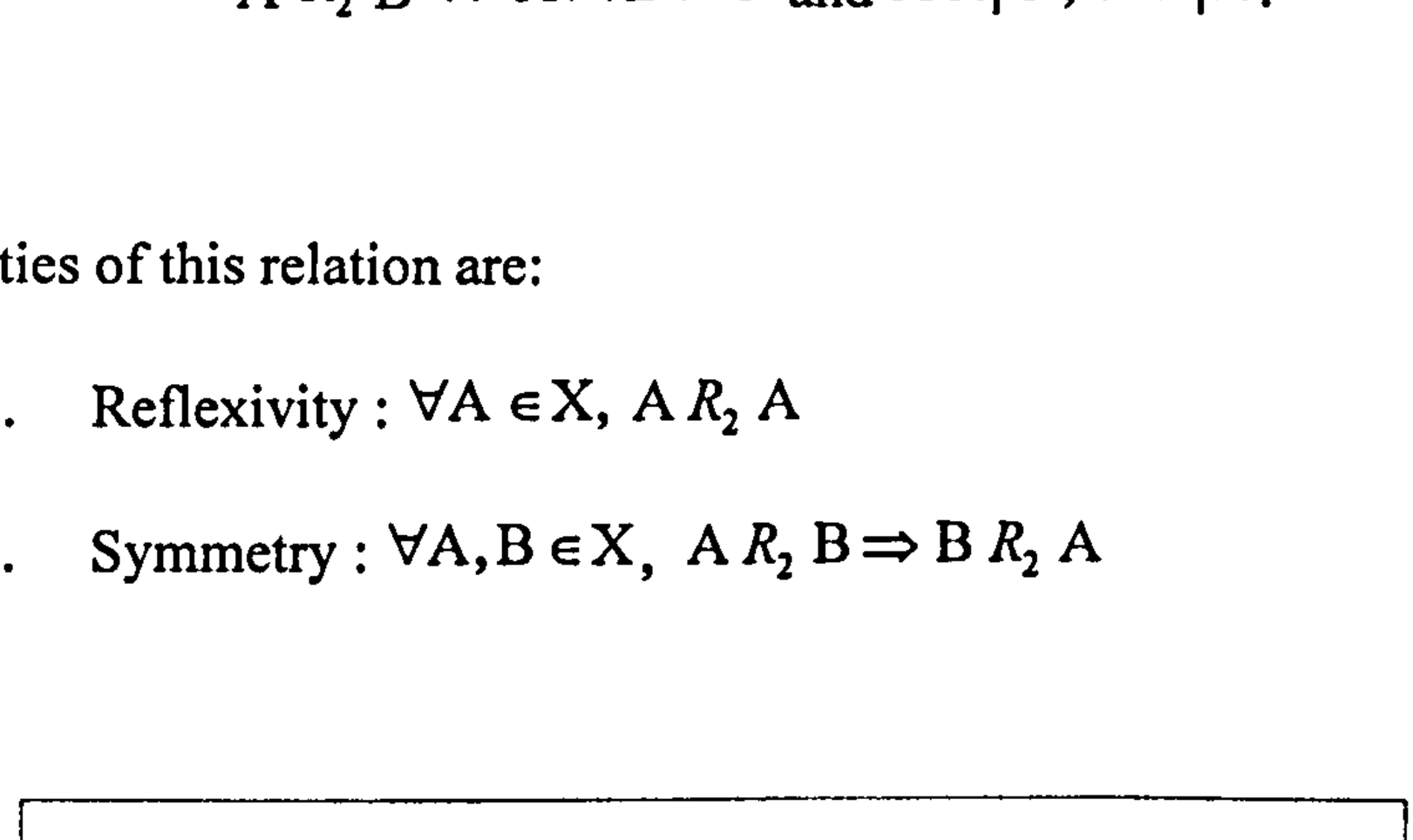


Figure 4.9 : The possible relations among the system cases

The overlapped relations between system cases can be further divided into two types based on the characteristics of the common elements of the overlapped cases. One is called *the concept level of the overlapped relation*, where the common elements of the overlapped cases are all in the concept level. For example, the two overlapped cases are case 7: {E1, E2, E3, E4, E5, R(E1,E2),

R(E1,E3), R(E1,E4), R(E4,E5)} and case 9: {E2, E3, R(E2,E6), R(E3,E7), R(E6,E7)}. The common elements of case 7 and case 9 are case 7 ∩ case 9 = {c2, c3} (from the viewpoint of the concept base), as shown in Figure 4.10. The definition of the concept level of the overlapped relation will be formalised in later section: the formalisation of the dual-base knowledge structure.

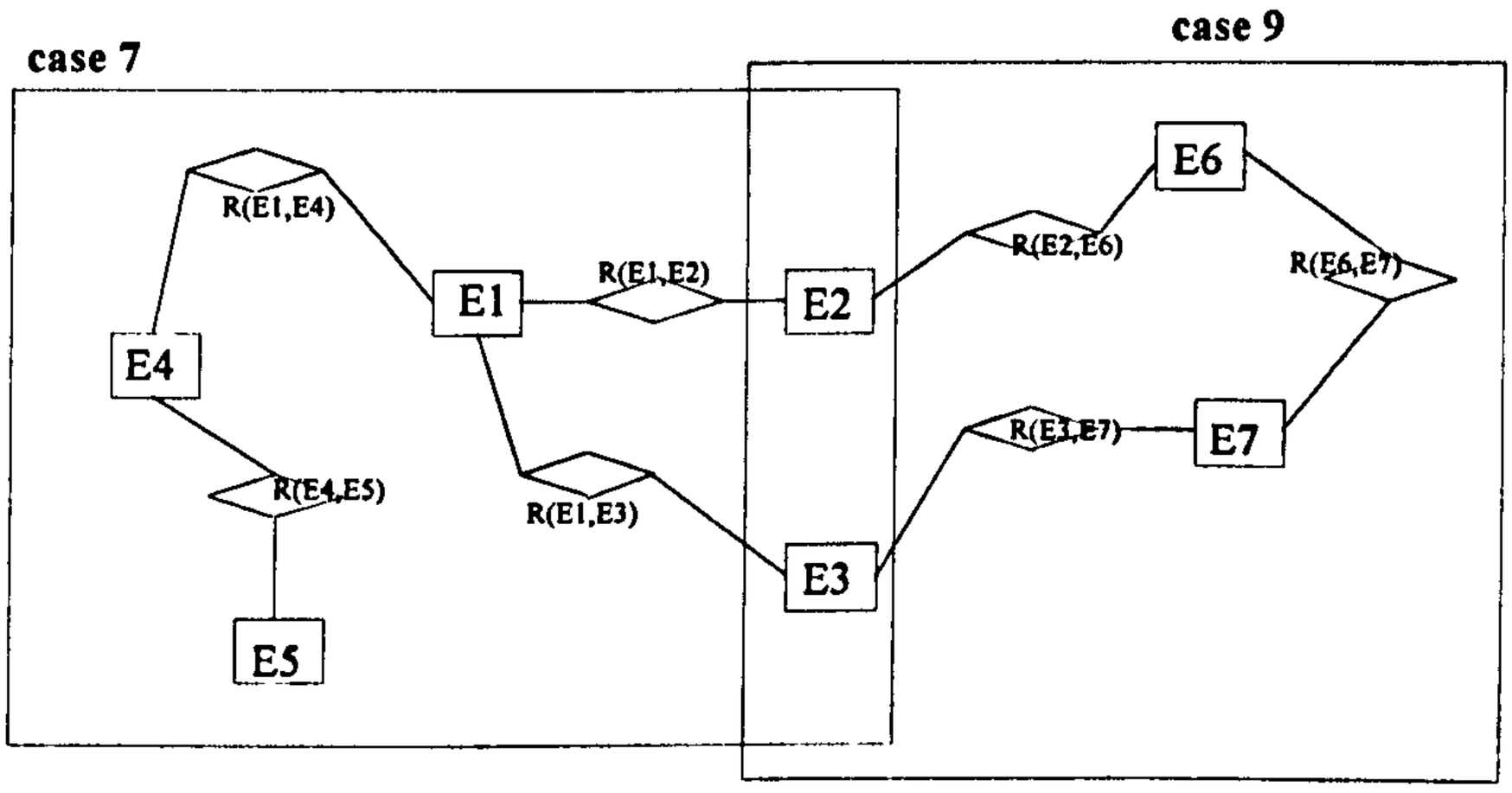


Figure 4.10 : The concept level of the overlapped relation

The second type is called *the concept-relation level of the overlapped relation*, where the common elements consist of the relations as well as the concepts. For example, the two overlapped cases are case 7: {E1, E2, E3, E4, R(E1,E2), R(E2,E3), R(E1,E4), R(E4,E3)} and case 9: {E1, E2, E3, E5, E6, R(E1,E2), R(E2,E3), R(E2,E5), R(E2,E6)}. The common elements of case 7 and case 9 are case 7 ∩ case 9 = {c1, c2, c3, r(c1,c2), r(c2,c3)} (from the viewpoint of the concept base), as shown in Figure 4.11. Thus, the definition of the concept-relation level of the overlapped relation will be formalised in later section: the formalisation of the dual-base knowledge structure.

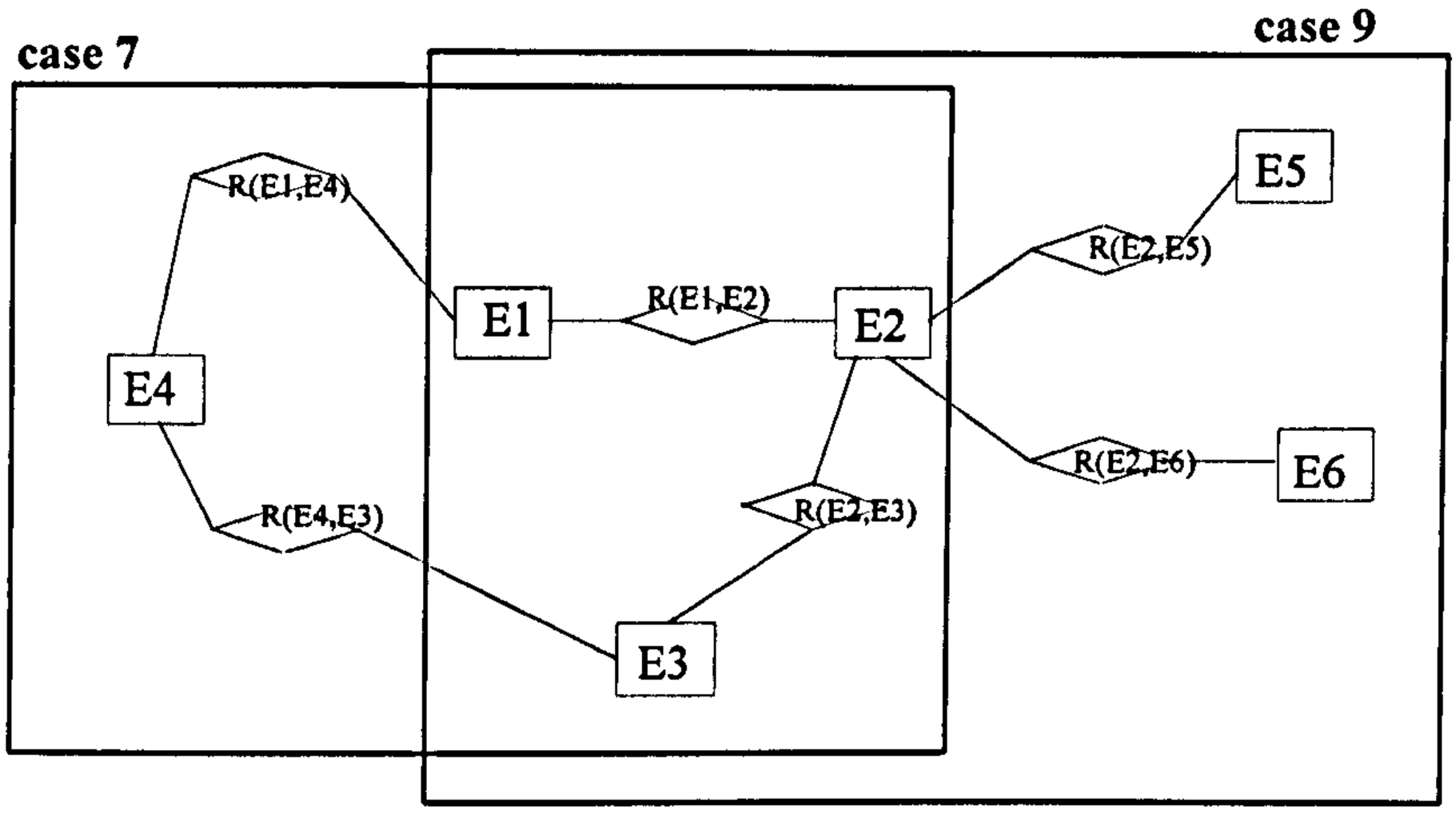


Figure 4.11 : The concept-relation level of the overlapped relation

4.2.3 The Relevance of the Concept and Case Bases

The concept base is formed by concepts and links, and the case base includes a number of interrelated cases. Although they are all representation systems for the referent, the real world, the ways in which they represent the objects and the relations among them are different. One is an abstract way (the concept base) and another is more specific (the cases), i.e. they are non-equivalent representations [Palmer, 1978]. Nevertheless, this phenomenon can be co-ordinated by our model from the micro and macro viewpoints as follows:

The micro view is concerned with the relevance of concepts and entities. The relationships between *concept* and *entity* are viewed from two perspectives. First, a concept is formed by an abstraction of *entity instances* in a world. Entity instances are extensions of concepts. Secondly, an *entity* in a world is regarded as a specific exemplar of a concept in the concept network. A concept in the concept

network may have more than one exemplar (entity) which appears in different applications. Exemplar links are used to represent the correspondence between concepts and entities. It is the exemplar links that establish the extensional meanings of concepts, as shown in Figure 4.12. In this example, a concept is assumed to have the same name as its exemplars (entities) in the case base[51]. Although a concept and its corresponding entities are represented by the same name, they may have different contents or play different roles in different applications. The relevance of the AUTHOR concept in the concept base and its exemplars, AUTHOR entity in case 9 and AUTHOR entity in case 7, is an example of this view.

The problems of conceptualisation are caused by the chaotic environment which the designers seek to understand and with which they interact. Thus, the characteristics of object concepts in the object identification and interpretation sub-phase may be summarised according to the reasons discussed in Chapter 2.. The typical exemplar of one concept is not universally true for all domains, i.e. the typical exemplar of one concept depends entirely on its intended used. For example, the AUTHOR concept in the library and bookstore domains is a case which illustrates this condition. The typical AUTHOR concept includes the name and profession as its features in the library domain, but it includes the name,

---

[51] It is not the case that the names of the exemplars of a concept must be same as the name of the concept. Even if the exemplar names of a concept are different (i.e. they are synonymous), they can be co-ordinated by their correspondence - their corresponding concept in the concept base.

address, sex, tel., fax as its features in the bookstore domain. They are different typical exemplars in different domains, but both are plausible and reasonable.

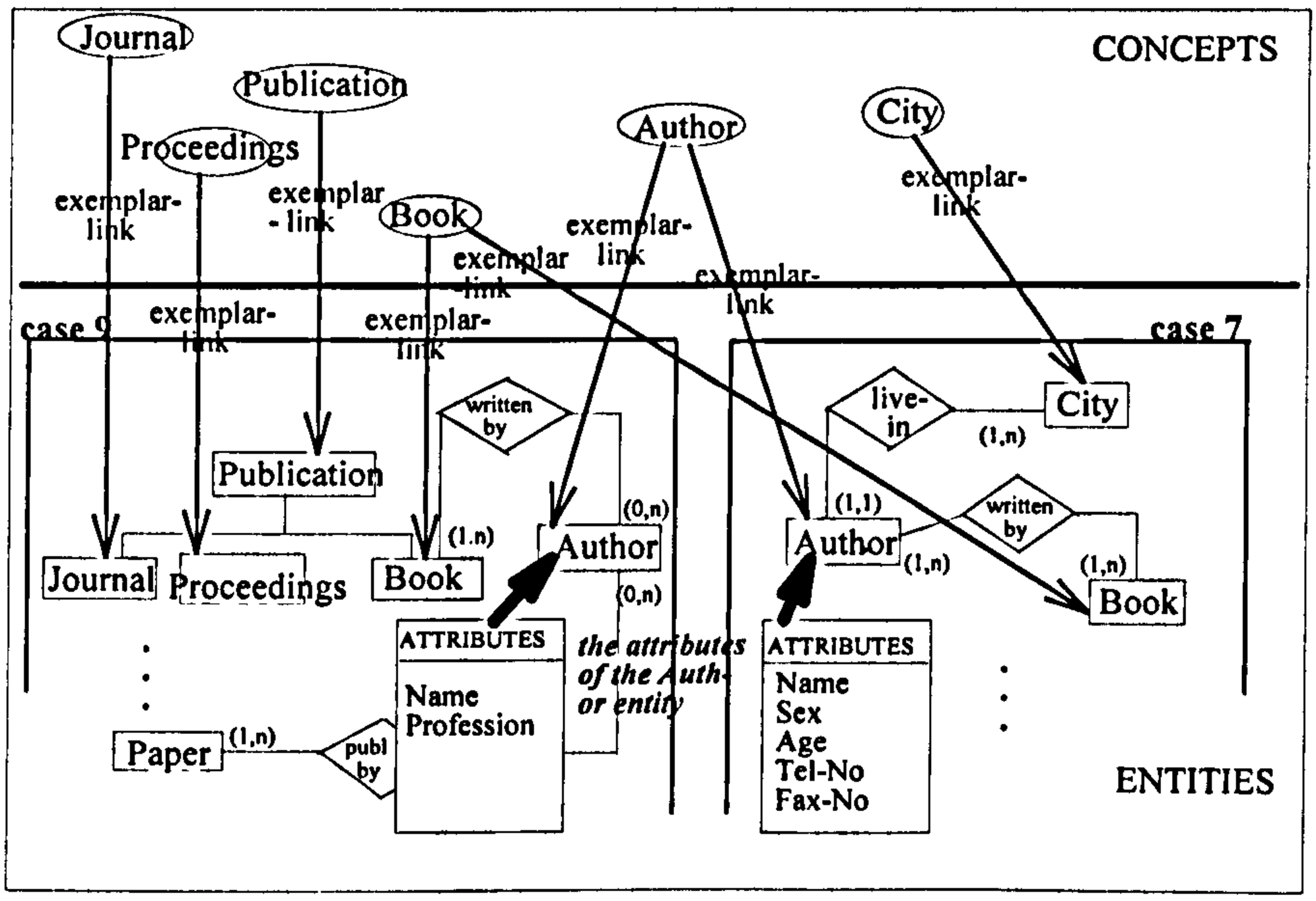


Figure 4.12 : The relevance of concepts and entities

The macro view is concerned with the correspondence between the concept and case base which is depicted as follows:

(1) Let set A include all the elements (entities and relationships) in the case base and set B include all the elements (concepts and relations) in the concept base, then there exists a *surjective* function from A to B (see Figure 4.13), denoted as follows: (The more precise definition will be formalised in the section of the fomalisation of the dual-base knowledge structure)

$$f: A \rightarrow B$$

$$\forall y \in B,\ \exists x \in A \quad s.t. \quad y = f(x)$$

i.e. every element in the concept base has at least one correspondence in the case base.

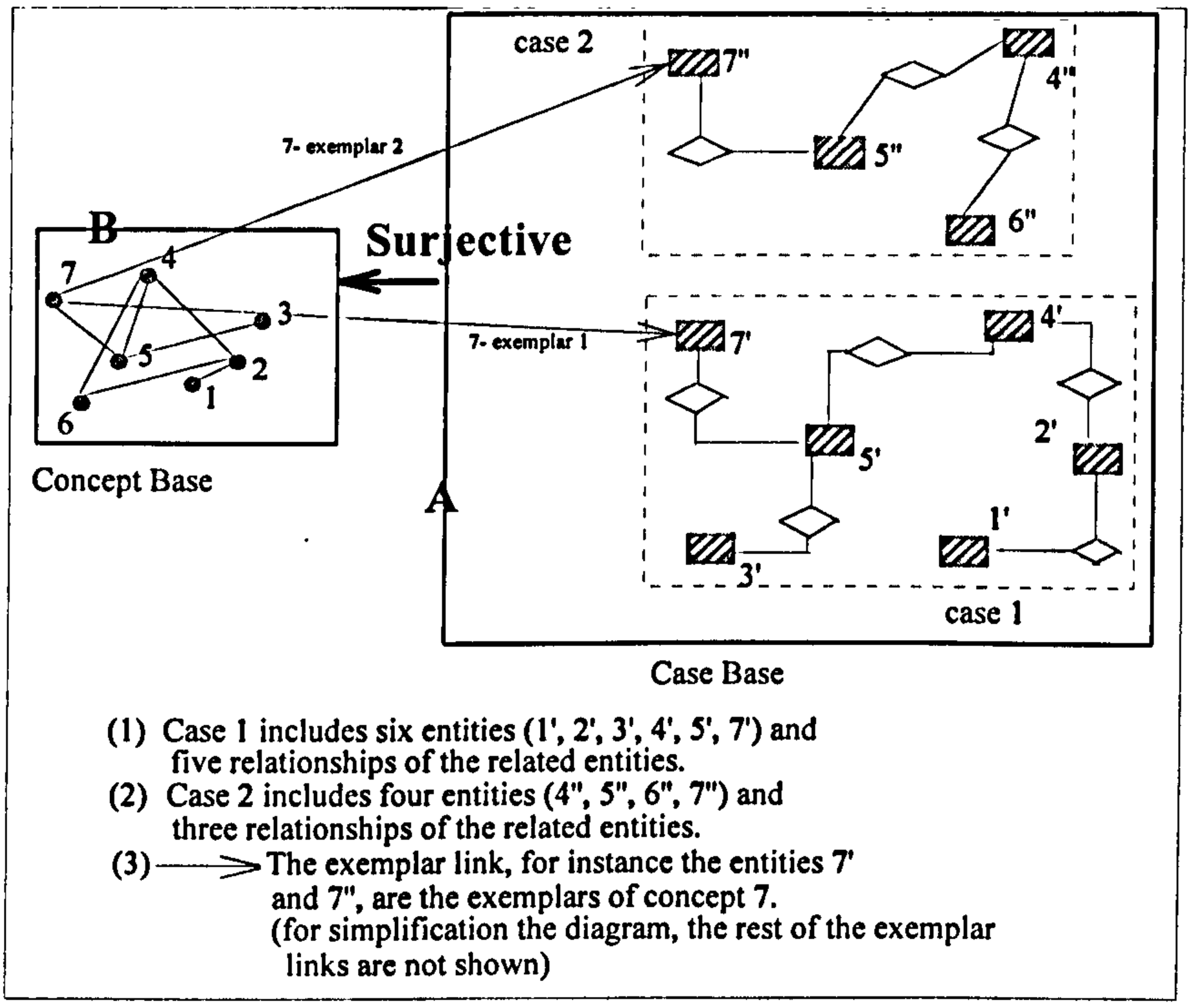


Figure 4.13 : A correspondence from the entity set (A) to the concept set (B)

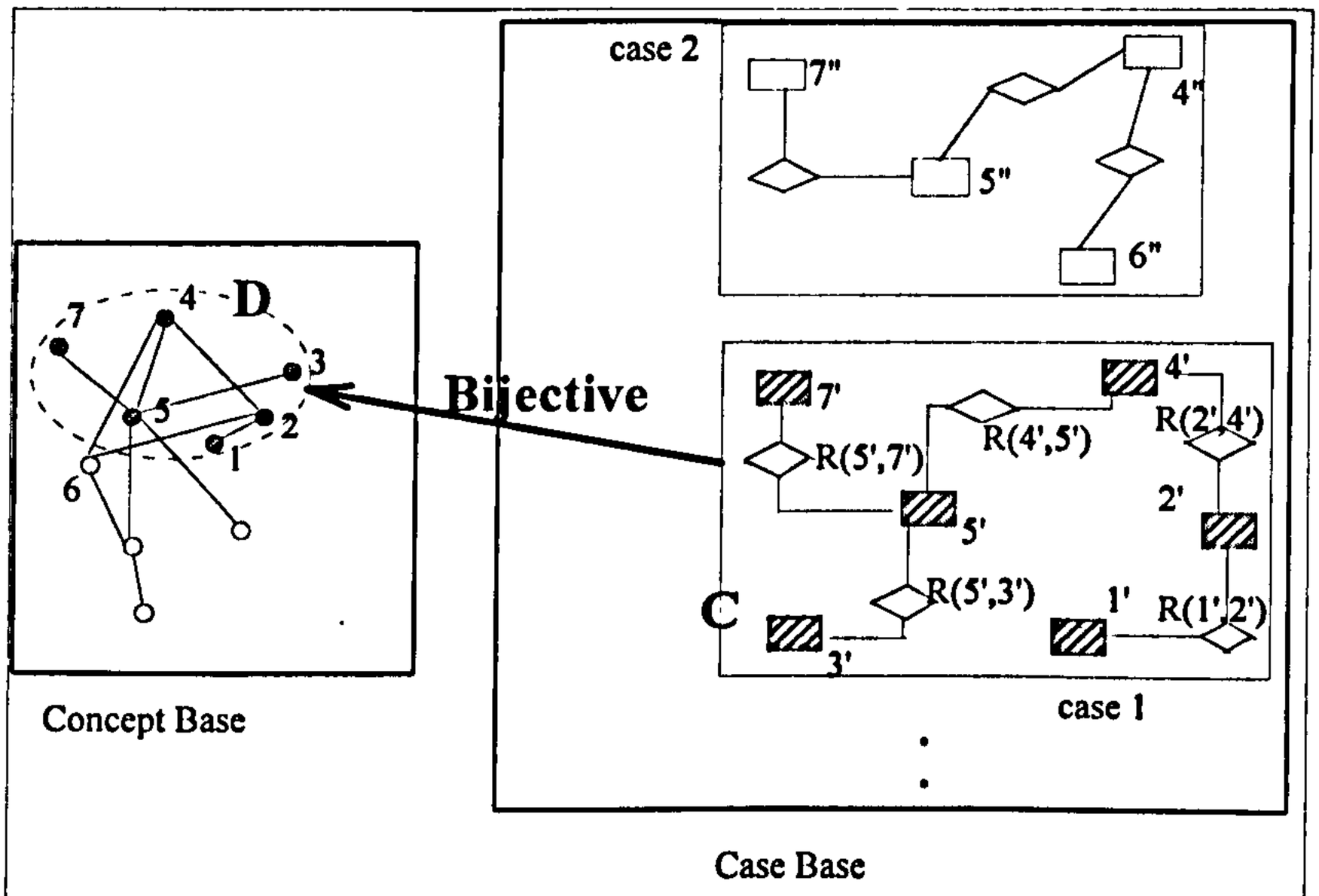
If the elements entities (relationships) in a case and their corresponding concepts (relations) in the concept base are regarded as two sets, namely C and D (the parts of the concept base) respectively, then

(2) an element in C and its corresponding element in D are associated in pairs, i.e. there exists a *bijective* function from C to D, and C and D are in *one-to-one correspondence* (see Figure 4.14), denoted as follows: (The more precise definition will be formalised in the section of the fomalisation of the dual-base knowledge structure)

$f: C \rightarrow D$

(1)if $(x_1, y) \in f$ *and* $(x_2, y) \in f$, *then* $x_1 = x_2$

$(2) \forall y \in D, \exists x \in C$ *s.t.* $y = f(x)$



(1) The case 1 includes six entities and five relationships of the related entities.

(2) The associated pairs from case 1 to its corresponding part of the concept base are:
1' - 1, 2' - 2, 3' - 3, 4' - 4, 5' - 5, 7' - 7;
R(1',2') - r(1,2), R(2',4') - r(2,4), R(4',5') - r(4,5), R(5',3') - r(5,3),
R(5',7') - r(5,7)

Figure 4.14 : A correspondence from an entity set (C) of a case to its correspondent concept set (D) in the concept base

4.3 The Properties of the Dual-Base Knowledge Model

4.3.1 The Duality of the Concept and Case Bases

Every concept in the concept base must have at least one corresponding entry in the case base. From the concept viewpoint, there are two types of exemplar in the case base, as illustrated in Figure 4.15. One is the case entity included in the case,

and the other is called the non-case entity that stands alone. This variation not only relaxes the condition that the exemplar of a concept must be of some cases, but also does not conflict with the model proposed above.

Because of this property, the confirmed concept(s) must have its (their) exemplar(s) in the case base.

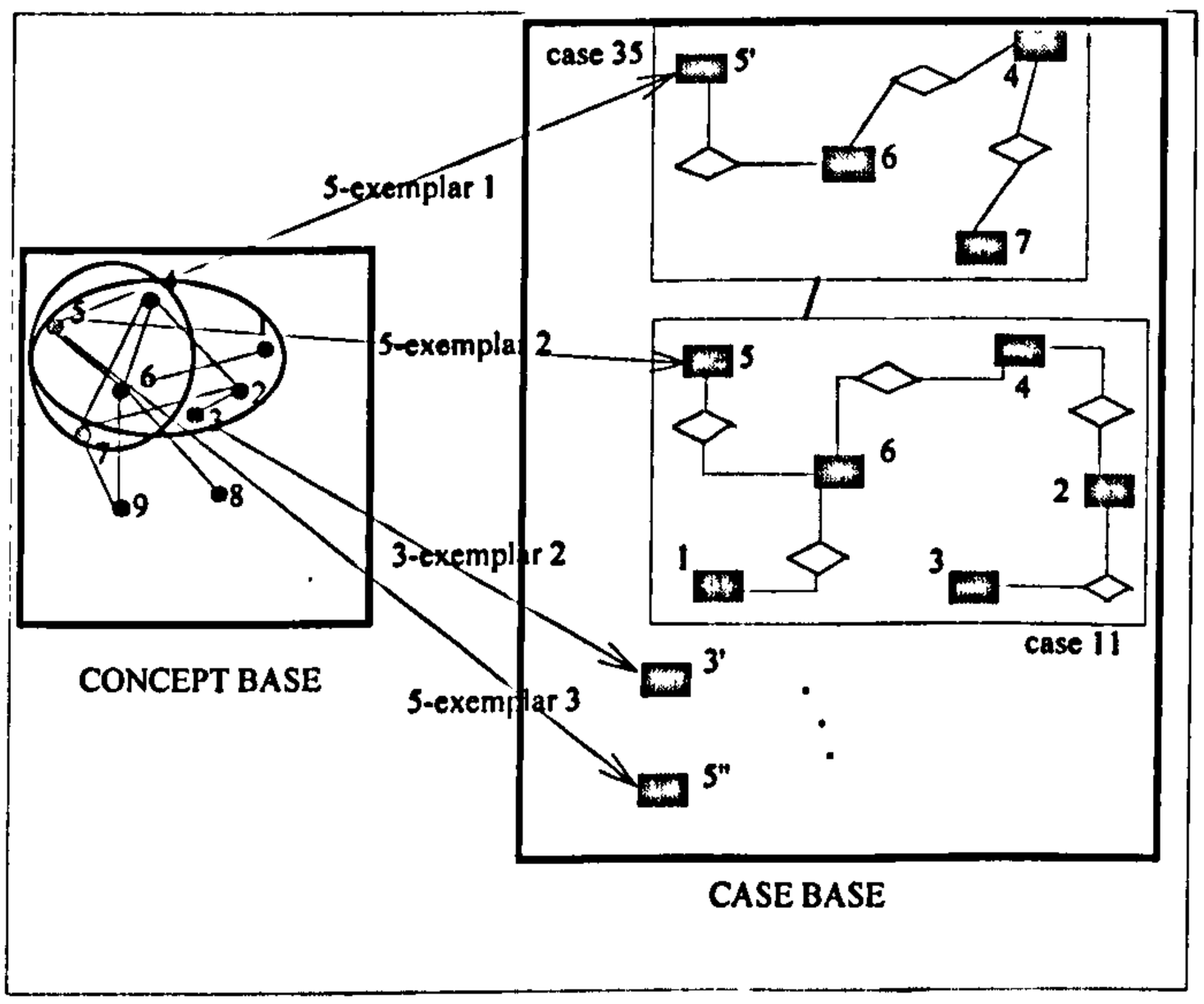


Figure 4.15 : The duality of the concept and case bases

4.3.2 General Domain Knowledge as an Inference Source

The concept base representing general domain knowledge can serve as a source for inference. The concept-links in the concept base can be used to check whether there are relations among concepts, thus implying that their exemplars exist in the case base. In other words, when a new entity is required but designers do not

know whether it is included in the very large and complicated case base, they can check whether there is a relation between the required concept and the concept which corresponds to an entity in the working case defined in next section, in the concept base, thus implying the required entity exists in some cases. This process can be guaranteed by virtue of the correspondence of the two bases described in section 4.2. For instance, if the working case includes four entities (4,5,6,7), the name of the new entity is 2 and the related entity of the new entity in the working case is 4. The process (in Figure 4.16) is that, first, concept 4 will be used as a basis to find whether there is a related concept that is not concept 5, 6 or 7. Secondly the name of the related concept, e.g. concept 2, will be displayed for confirmation by the designers, and finally the confirmed concept (concept 2) will be packed and transferred to the case base as index to find its exemplars (entities) that are included in some cases.

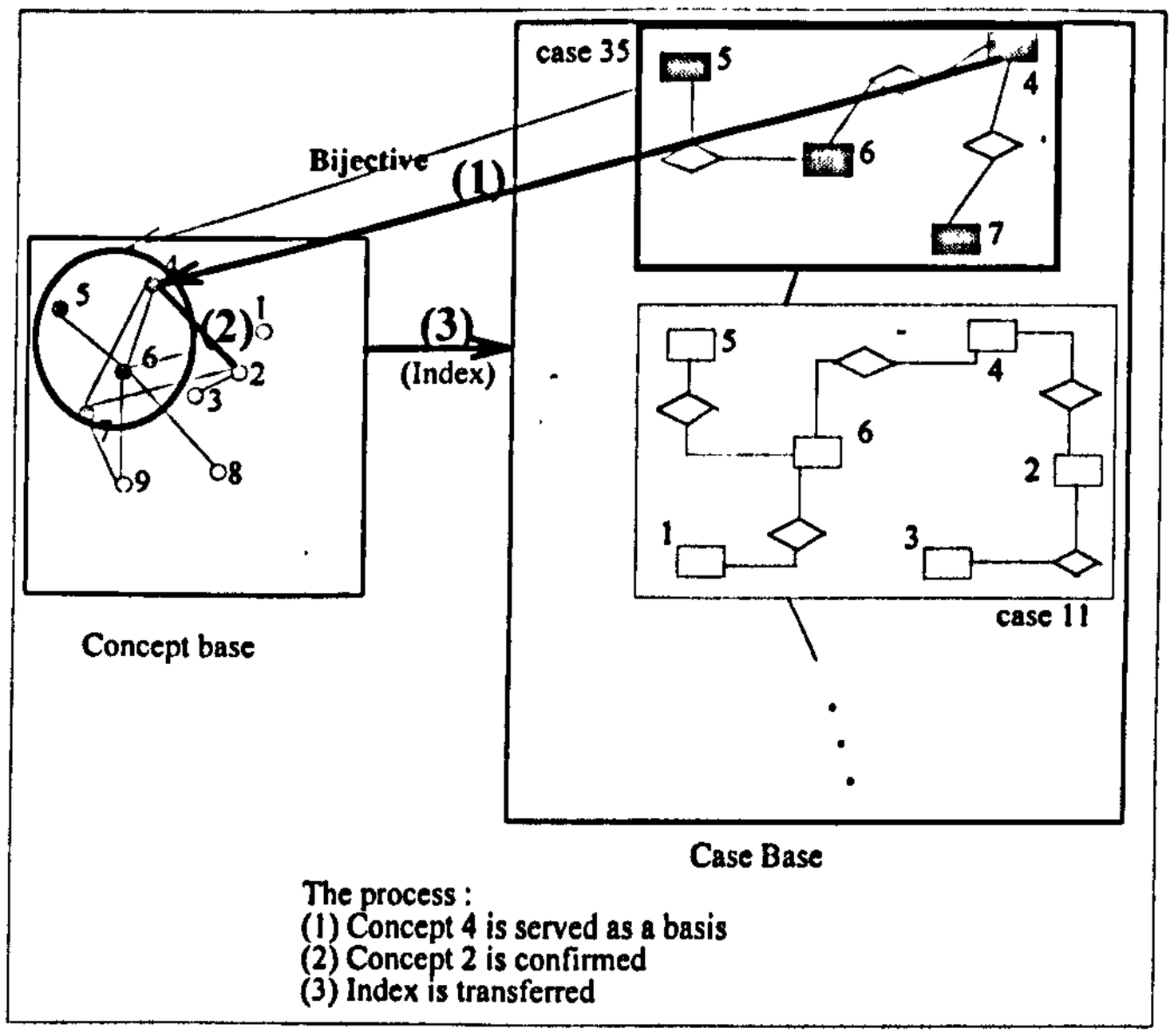


Figure 4.16 : The inference process

4.3.3 Non-symmetry Degree and Variant Similarity of the Overlapped Cases

The degree and similarity of overlapped cases are two factors influencing the retrieval of cases. Their characteristics are discussed briefly below.

4.3.3.1 The overlapped degree is non-symmetric and directional

Although the overlapped relations among cases are symmetrical, the degree of overlapped case may be not symmetrical. The degree from A to B does not necessarily equal the degree from B to A, i.e. the comparison is directional. The degree of two overlapped cases (assuming A is the subject and B is the referent) is expressed as a function of three arguments: $A \cap B$, the common element(s) of A and B; $A - B$, the elements that belong to A but not to B; $B - A$, the elements that belong to B but not to A. Thus, the degree of overlapped cases is determined by the number of unique elements in each case, as well as by the number of elements in common. In this research, the ratio model [Tversky, 1977] is adopted and modified to calculate the degree, represented as follows :

$$D_{overlapped}(A,B) = \frac{f_A(A \cap B)}{f_A(A \cap B) + f_A(A - B) + f_B(B - A)},$$

where

$D_{overlapped}(A,B)$: *overlapped degree from A to B*

$$f_A(A \cap B) = \frac{\#(A \cap B)}{\#A}, f_A(A - B) = \frac{\#(A - B)}{\#A},$$

$$f_B(B - A) = \frac{\#(B - A)}{\#B};$$

$\#(X)$: *Number of elements of X*

4.3.3.2 The similarity is relative and variable

Kolodner and Wills [1993] suggested that anticipatory indexing is not sufficient fully to explain retrieval. Features that were not salient at the time a case was experienced might be important for retrieval in the current situation. The overlapped relations among cases (see Figure 4.9) imply that any entity in a case can be included in the finding probe (index). It is always true that a case will be retrieved in a different way, i.e. the case base is content addressable in that any entities of a case can potentially serve as an index for access to the entire case. Then, the overlapped cases can be viewed as either similar or different to the referent case, depending on the users' requirements (the content of the index).

The two characteristics can be used as a criterion for selecting. Which overlapped cases are similar to the referent depends, first, on the content of the index, and then on the overlapped degree from these cases to the referent case.

4.3.4 The Grade Structure of the Case Base

The notion of the grade structure of the case base is influenced by the theory of best-example (i.e. the typicality of examples) explored by many psychologists [Rosch and Mervis, 1975; Rosch, 1975a, 1978]. Mervis and Pani [1980] suggested that using best examples is easier and more accurate than using poor examples in people generalisation and learning. In addition, the best example

serves as a *reference point* for inference and decision-making [Cherniak, 1984; Rosch, 1975b; Rips, 1975].

In this research, the role of the typical case is twofold: (1) to provide a holistic comprehension of the working domain; (2) to serve as a reference point for the recalling process.

4.4 The Formalisation of the Dual-Base Knowledge Structure

The DBKM includes two synergistic knowledge structures: the concept and case bases. They complement each other and work together to retrieve relevant cases. In this section, the general representations of these two structures and their relevance are formalised by set theory as follows.

4.4.1 The Formalisation of the Concept Base

According to the structure of the concept base discussed in section 4.2.1, the concept base can be defined as follows.

Definition: B is the concept base. Let B be defined as a pair of sets as follows:

$$B = (CB, RC)$$

where $CB = \{c_1, c_2, \ldots, c_n\}$

c_i : the object concept

$RC = \{r1(c_{a1}, c_{b1}), r1(c_{a2}, c_{b2}), \ldots, r2(c_{x1}, c_{y1}), r2(c_{x2}, c_{y2}), \ldots\}$ including only two types of relation of object concepts,

r1(...): the undefined relation (U_D) of related object concepts

$r2(c_{xi}, c_{yi})$: the IS-A relation (IS-A) of the c_{xi} and c_{yi} object concepts, i.e. the c_{yi} object concept is a generalisation of the c_{xi} object concept.

According to the above definition, an example of the concept base, as shown in Figure 4.17, can be written as follows.

Concept base =

({a,b,c,d,e,f,r,w,y,z},

{r1(a,b),r1(a,c),r1(a,e),r1(a,r),r1 (b,d),r1(d,f),r2(w,a),r2(y,a),r2(z,a)})

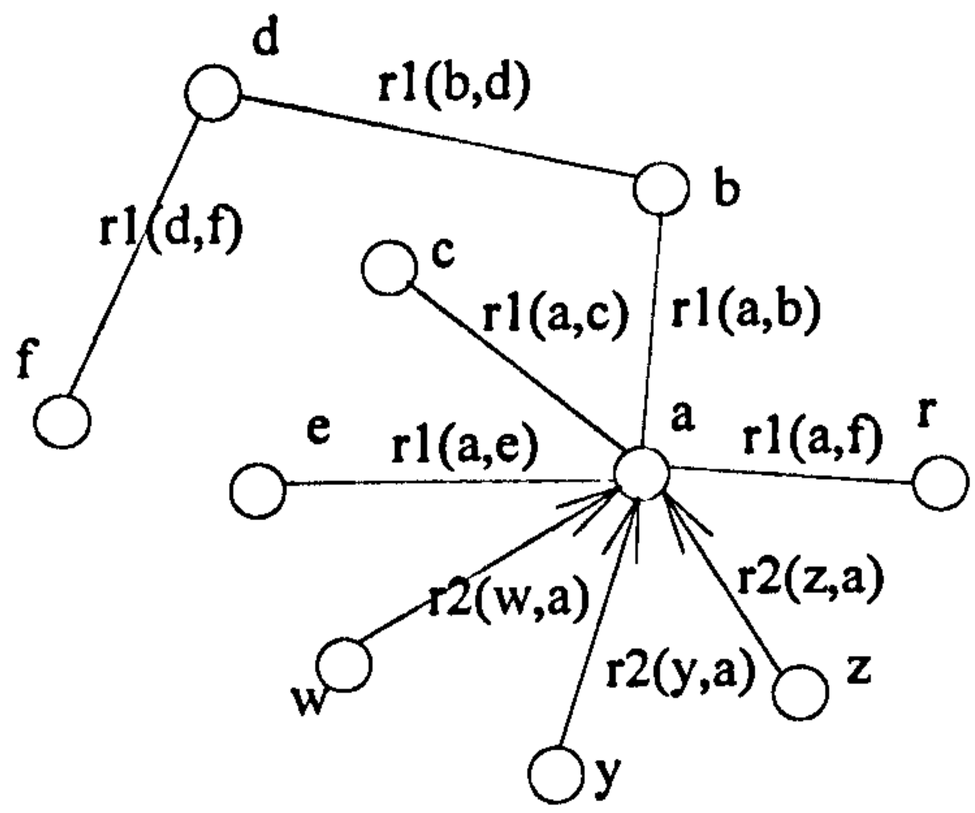


Figure 4.17 : An example of the concept base

4.4.2 The Formalisation of a Case

According to the structure of the case base discussed in section 4.2.2, a case can be defined as follows.

Definition: A is a case. Let A be defined as a pair of sets as follows.

$$A = (EA, RE)$$

where $EA = \{e_1, e_2, ..., e_n\}$

e_i: the entity

$RE = \{R1_1(e_{a1}, e_{b1}), R1_2(e_{a2}, e_{b2}), ..., R2(e_{x1}, e_{y1}), R2(e_{x2}, e_{y2}), ...\}$

$R1_i$(....): the relationship of the related entities.

$R2(e_{xi}, e_{yi})$: the IS-A relationship of the e_{xi} and e_{yi} entities, i.e. the e_{yi} entity is a generalisation of the e_{xi} entity.

According to the above definition, an example of a case, as shown in Figure 4.18, can be written as follows.

Case =

({ BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER},
{IS-A(BOOK,PUBLICATION), IS-A(PROCEEDINGS,PUBLICATION),
IS-A(JOURNAL,PUBLICATION), Written_by(PAPER,AUTHOR),
Writes_Books(BOOKS,AUTHOR), Belongs_to_Jour.(PAPER,JOURNAL),
Belongs_to_Proc.(PAPER,PROCEEDINGS)})

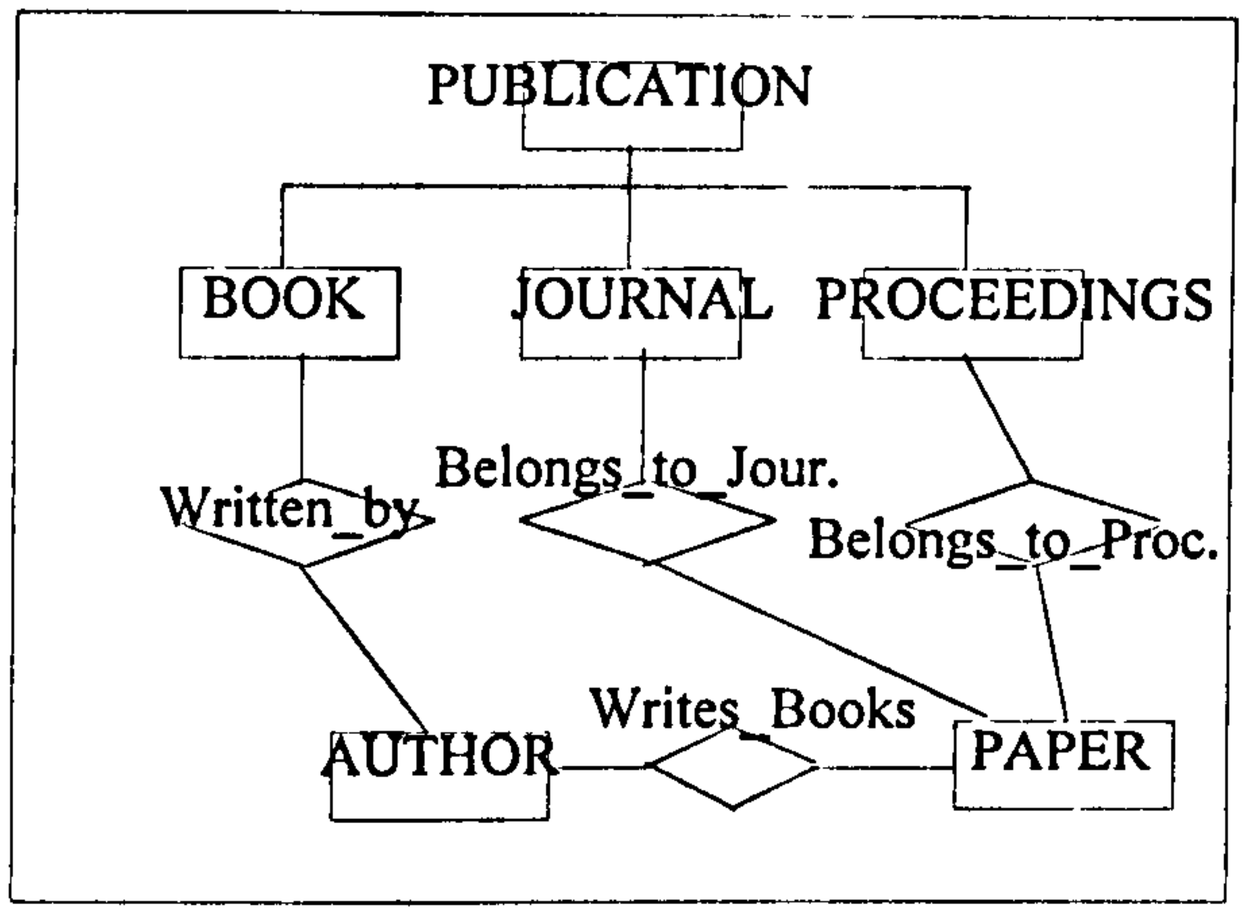


Figure 4.18 : An example of a case

4.4.3 The Formalisation of the Relevance of the Concept and Case Structures

According to the relevance of the concept and case bases discussed in section 4.2.3 and the duality of the concept and case bases discussed in section 4.3.1, the internal representations of the relevance of the non-equivalent representation structures can be formalised by the above definitions of the concept and case bases.

Definition: $A_i = (EA_i, RE_i)$ and $SE = \{e_{s1}, e_{s2}, \ldots\}$ is a set of non-case (stand alone) entities in the case base. Let D be a set of the elements in the case base and be defined as follows.

$$D = \bigcup_i EA_i \cup RE_i \cup SE$$

$$= \{EA_1 \cup EA_2 \cup \ldots \cup EA_n \cup SE\} \cup \{RE_1 \cup RE_2 \cup \ldots \cup RE_n\}$$

$$= \{e_{11}, e_{21}, \ldots, e_{m_1 1}\} \cup \{e_{12}, e_{22}, \ldots, e_{m_2 2}\} \cup \ldots \cup \{e_{1n}, e_{2n}, \ldots, e_{m_n n}\} \cup \{e_{s1}, e_{s2}, \ldots\} \cup$$

$$\{R1_{11}(e_{a11}, e_{b11}), R1_{21}(e_{a21}, e_{b21}), \ldots, R2_1(e_{x11}, e_{y11}), R2_1(e_{x21}, e_{y21}), \ldots\} \cup$$

$$\{R1_{12}(e_{a12}, e_{b12}), R1_{22}(e_{a22}, e_{b22}), \ldots, R2_2(e_{x12}, e_{y12}), R2_2(e_{x22}, e_{y22}), \ldots\} \cup \ldots \cup$$

$$\{R1_{1n}(e_{a1n},e_{b1n}),R1_{2n}(e_{a2n},e_{b2n}),\ldots,R2_{n}(e_{x1n,}e_{y1n}),R2_{n}(e_{x2n},e_{y2n}),\ldots\}$$

$$= \{e_{11},e_{21},\ldots,e_{m_1 1},e_{12},e_{22},\ldots,e_{m_2 2},\ldots,e_{1n},e_{2n},\ldots,e_{m_n n},e_{s1},e_{s2},\ldots,$$

$$R1_{11}(e_{a11},e_{b11}),R1_{21}(e_{a21},e_{b21}),\ldots,R2_{1}(e_{x11,}e_{y11}),R2_{1}(e_{x21},e_{y21}),\ldots,$$

$$R1_{12}(e_{a12},e_{b12}),R1_{22}(e_{a22},e_{b22}),\ldots,R2_{2}(e_{x12,}e_{y12}),R2_{2}(e_{x22},e_{y22}),\ldots,\ldots\ldots,$$

$$R1_{1n}(e_{a1n},e_{b1n}),R1_{2n}(e_{a2n},e_{b2n}),\ldots,R2_{n}(e_{x1n,}e_{y1n}),R2_{n}(e_{x2n},e_{y2n}),\ldots\}$$

The exemplar link of a concept and its corresponding entity can be defined as follows.

$$\text{has-exemplar}(c_i, e_{iki\ (or\ si)})$$

where c_i : a concept

$e_{iki\ (or\ si)}$: the corresponding entity e_i of the concept c_i in case k or in SE

$B_i = \{CB_i, RC_i\}$ is a set of the corresponding elements of the entities (relationships) of case i. The bijective function from A_i to B_i can be formalised as follows.

$$f_i: A_i \rightarrow B_i$$

(1)if $(e_i,c_i) \in f_I$ and $(e_j,c_i) \in f_I$, then $e_i = e_j$;

if $(R1_i,r1_i) \in f_1$ and $(R1_j,r1_i) \in f_1$, then $R1_j = r1_i$;

if $(R2_i,r2_i) \in f_1$ and $(R2_j,r2_i) \in f_1$, then $R2_j = r2_i$

(2)$\forall c_i \in CB_i$, $\exists e_i \in EA$ s.t. $c_i = f_I(e_i)$;

$\forall r1_i \in RC_i$, $\exists R1_i \in RE_i$ s.t. $r1_i = f_1(R1_i)$;

$\forall r2_i \in RC_i$, $\exists R2_i \in RE_i$ s.t. $r2_i = f_1(R2_i)$

The surjective function from D to B = (CB,RC) can be formalised as follows.

$$f_2: D \rightarrow B$$

$$\forall c_i \in CB, \exists e_{iki\ (or\ si)} \in D \text{ s.t. } c_i = f_2\ (e_{iki\ (or\ si)});$$

$$\forall r1 \in RC, \exists R1_{ij} \in D \text{ s.t. } r1 = f_1(R1_{ij});$$

$$\forall r2 \in RC, \exists R2_{ij} \in D \text{ s.t. } r2 = f_1(R2_{ij})$$

The example of the above definition, as shown in Figure 4.19, can be represented as follows.

(1) B = {1,2,3,4,5,6,7, r1(1,6), r1(2,3), r1(2,4), r1(4,6), r1(5,6), r1(4,7)}

(2) EA_1 = {1,2,3,4,5,6} (case 1), EA_2= {4,5',6,7}(case 2) and SE = {3',5''}

(2) RE_1 = { $R1_{11}$(1,6), $R1_{21}$(2,3), $R1_{31}$(2,4), $R1_{41}$(4,6), $R1_{51}$(5,6)}(case 1),

RE_2 = { $R1_{12}$(5',6), $R1_{22}$(4,7), $R1_{32}$(4,6), }(case 2)

(3) $D = \bigcup_i EA_i \cup RE_i \cup SE = EA_1 \cup RE_1 \cup EA_2 \cup RE_2 \cup \{3',5''\}$

$= \{1_1,2_1,3_1,4_1,5_1,6_1,4_2,5'_2,6_2,7_2,3'_{S1},5''_{S2}, R1_{11}(1,6), R1_{21}(2,3), R1_{31}(2,4), R1_{41}(4,6), R1_{51}(5,6), R1_{12}(5',6), R1_{22}(4,7), R1_{32}(4,6)\}$

(4) The exemplar links:

For the concept 1: has-exemplar$(1,1_1)$

For the concept 2: has-exemplar$(2,2_1)$

For the concept 3: has-exemplar$(3,3_1)$, has-exemplar$(3,3'_{s3})$

For the concept 4: has-exemplar$(4,4_1)$, has-exemplar$(4,4_2)$

For the concept 5: has-exemplar$(5,5_1)$, has-exemplar$(5,5'_2)$, has-exemplar$(5,5''_{s5})$

For the concept 6: has-exemplar$(6,6_1)$, has-exemplar$(6,6_2)$

For the concept 7: has-exemplar$(7,7_2)$

(5) The bijective function:

B_1 is a set of the corresponding elements of the entities (relationships) of A_1, then

$f_1: A_1 \rightarrow B_1$

where

$A_1 = \{1,2,3,4,5,6, R1_{11}(1,6),\ R1_{21}(2,3), R1_{31}(2,4), R1_{41}(4,6), R1_{51}(5,6)\}$;

$B_1 = \{1,2,3,4,5,6,\ r1_1(1,6),\ r1_1(2,3),\ r1_1(2,4),\ r1_1(4,6),\ r1_1(5,6)\}$

B_2 is a set of the corresponding elements of the entities (relationships) of A_2, then

$f_1: A_2 \rightarrow B_2$

where

$A_2 = \{4,5',6,7,\ R1_{12}(5',6),\ R1_{22}(4,7),\ R1_{32}(4,6)\}$;

$B_2 = \{4,5,6,7,\ r1_2(5,6),\ r1_2(4,7),\ r1_2(4,6)\}$

(6) The surjective function:

$f_2: D \rightarrow B$

where

$D = \{1_1,2_1,3_1,4_1,5_1,6_1,4_2,5'_2,6_2,7_2,3'_{S1},5''_{S2},\ R1_{11}(1,6),\ R1_{21}(2,3),\ R1_{31}(2,4),$

$R1_{41}(4,6),\ R1_{51}(5,6),\ R1_{12}(5',6),\ R1_{22}(4,7),\ R1_{32}(4,6)\}$;

$B = \{1,2,3,4,5,6,7, r1(1,6), r1(2,3), r1(2,4), r1(4,6), r1(5,6), r1(4,7)\}$

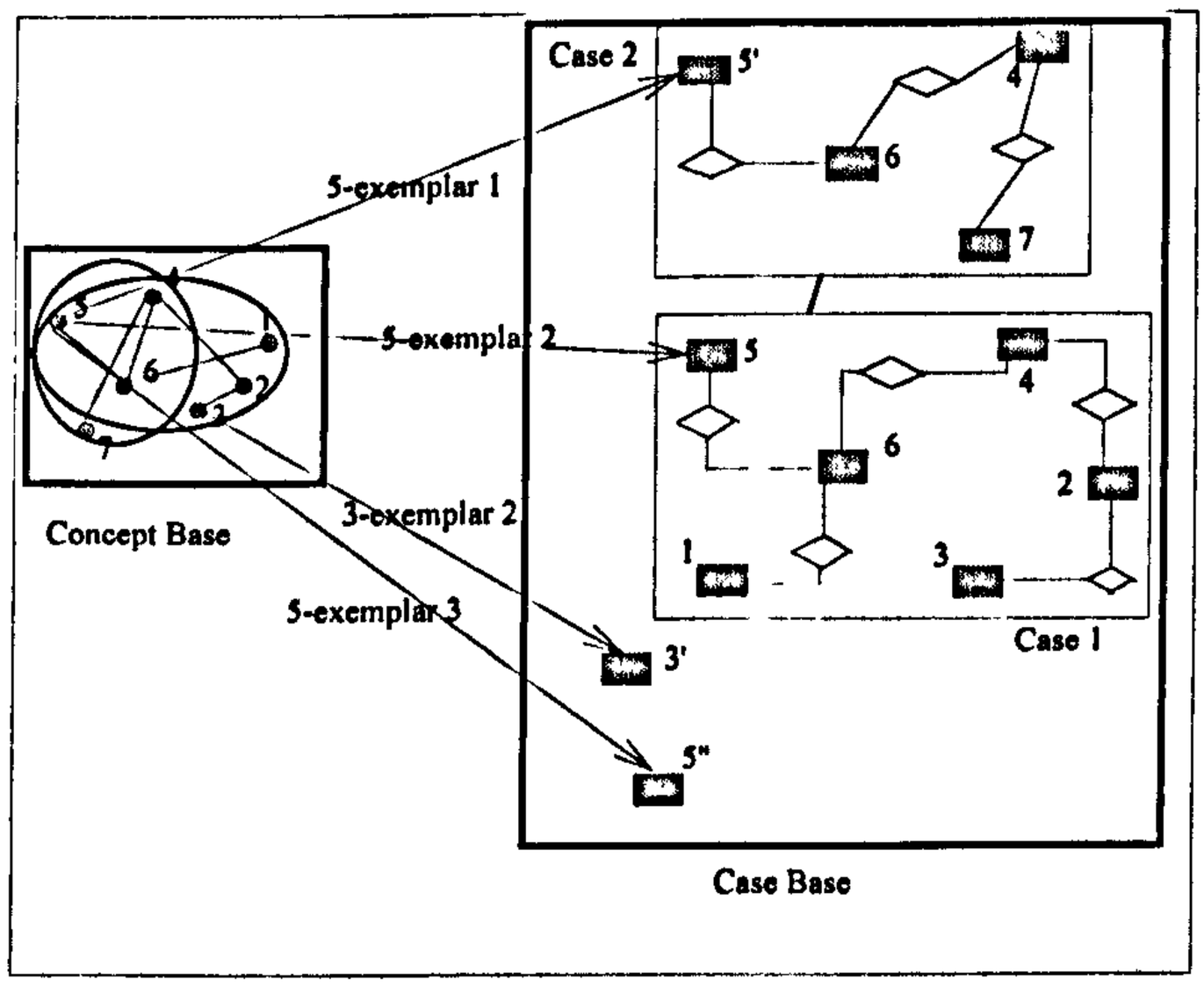


Figure 4.19 : An example of the concept and case bases

4.4.4 The Formalisation of the Relations of the Case Base

As discussed in section 4.2.2.2, two types of case - domain and system cases - are organised into the case base for different purposes. The former type is for analogous reference, and the latter type is for similar reasoning. Thus, based on the property of the domain cases: generalised from two or more system cases in same domain, the relation between the domain case and its system cases can be defined as follows.

Definition: A_{dci} is the domain case in the i domain. $A_{t\text{-}sc_j, j, i}$ is a typical system case of the j system in the i domain (the notion of multiple typical system cases of a domain case is like the notion of multiple exemplars [such as bird, mammal, fish] of a superordinate concept [such as animal], as discussed in section 3.2). Let is-typical be a link of the domain case and its typical case of one system and be defined as the following form.

$$\text{is-typical}(A_{t\text{-}sc_j, j, i}, A_{dci})$$

According to the above definition, an example of the relation between a domain case and its typical system cases, as shown in Figure 4.20, can be written as follows.

The is-typical links of the transportation control domain:

is-typical(Atc1, Adc1)

is-typical(Atc2, Adc1)

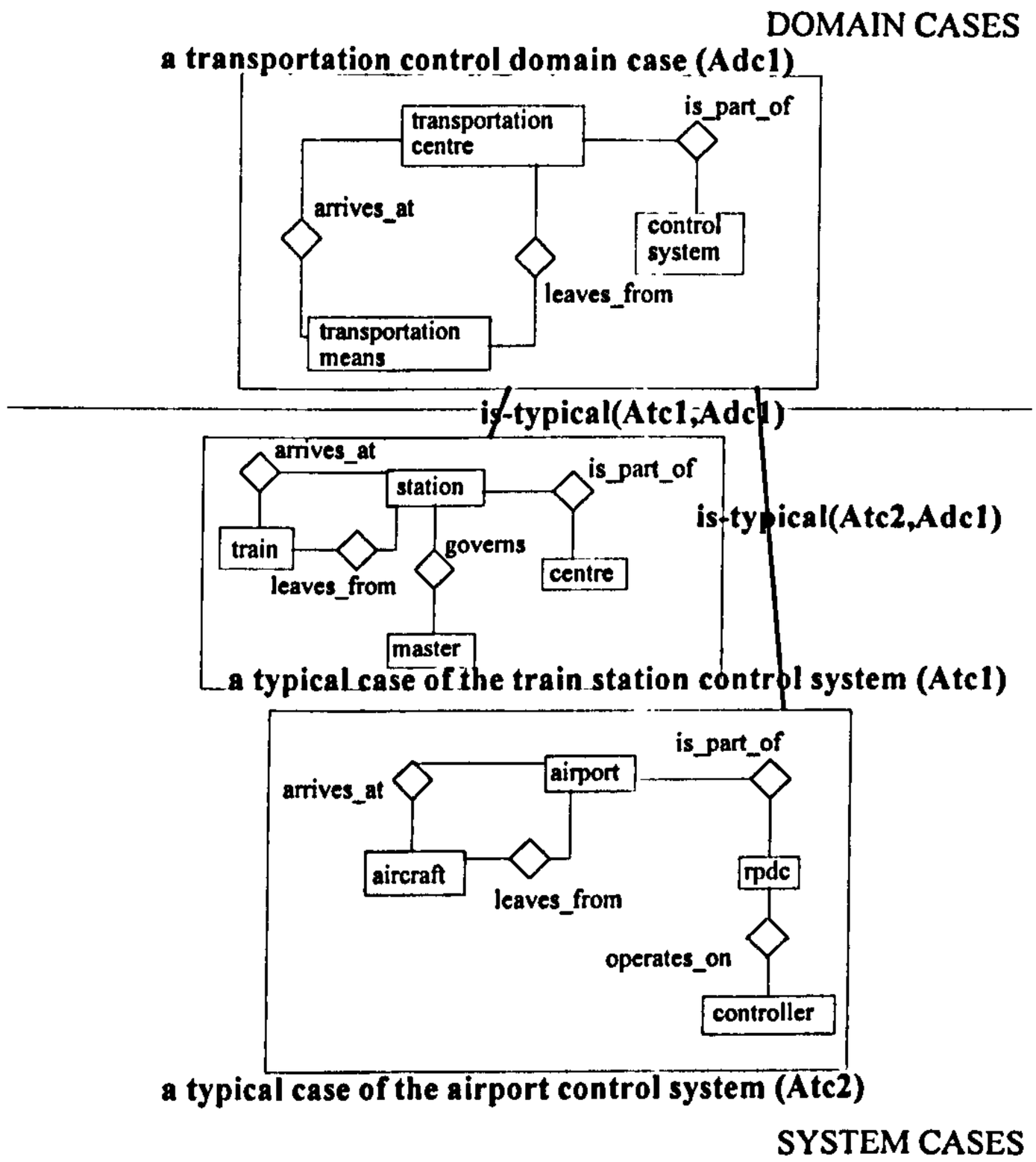


Figure 4.20 : An example of the relation between a domain case and its typical system cases

The issue of the relating system cases is concerned with the indexing cases. Therefore, the content of the relations among system cases, according to the discussion in section 3.3.2.1, must be concrete enough to be recognised and abstract enough to make a system case useful in a variety of future situation. Based on the flexibility of the concept base (derived from the notion of mental representation discussed in section 4.1.1) and the corresponding properties of the concept and case bases (derived from the notion of the correspondence between mental model and mental representation discussed in section 4.1.1), the content of a system case, from the viewpoint of the concept base, consists of the corresponding concepts (relations) of the system case. For example, in Figure 4.18, the content of the system case can be

viewed as a pair of sets: a set of the corresponding concepts and a set of the corresponding relations between (among) related corresponding concepts, as follows.

({ BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER},
{r2(BOOK,PUBLICATION), r2(PROCEEDINGS,PUBLICATION),
r2(JOURNAL,PUBLICATION), r1(PAPER,AUTHOR), r1(BOOKS,AUTHOR),
r1(PAPER,JOURNAL), r1(PAPER,PROCEEDINGS)})

Therefore, from the viewpoint of the concept base, two types of relation among system cases - the proper subset-superset and overlapped relations - discussed in section 4.2.2.2, can be formalised as follows.

Definition: $A_{sci} = (EA_{sci}, RE_{sci})$ and $A_{scj} = (EA_{scj}, RE_{scj})$ are two different system cases. According to the above discussion, A_{sci} can be viewed as a (CB_{sci}, RC_{sci}) pair set and A_{scj} can be viewed as a (CB_{scj}, RC_{scj}) pair set.

Let R_1 be the proper subset-superset relation of A_{sci} and A_{scj}. The relation can be defined as follows.

$$R_1(A_{sci}, A_{scj}) \Leftrightarrow A_{sci} \subset A_{scj} \text{ (i.e. } CB_{sci} \subseteq CB_{scj} \text{ and } RC_{sci} \subset RC_{scj})$$

Let R_2 be the overlapped relation of A_{sci} and A_{scj}. The relation can be defined as follows.

$$R_2(A_{sci}, A_{scj}) \Leftrightarrow A_{sci} \cap A_{scj} \neq \Phi \text{ and } R_1(A_{sci}, A_{scj}),\ R_1(A_{scj}, A_{sci})$$

As discussed in section 4.2.2.2, there are two types of the overlapped relation: the concept level (abbreviated as $overlp_{cl}$) and the concept-relation level (abbreviated as $overlp_{c\text{-}rl}$) of the overlapped relation. They can be defined as follows.

The definition of the concept level of the overlapped relation of A_{sci} and A_{scj}

$$overlp_{cl}(A_{sci}, A_{scj}) \Leftrightarrow R_2(A_{sci}, A_{scj}) \text{ and } CB_{sci} \cap CB_{scj} \neq \Phi,\ RC_{sci} \cap RC_{scj} = \Phi$$

According to the above definition, an example of this type of overlapped relation, as shown in Figure 4.21, can be represented as follows.

A_1 can be viewed (from the concept base viewpoint) as (CB_1, RC_1) =
({c1,c2,c3,c4,c5},{r1(c1,c2),r1(c1,c3),r1(c1,c4),r1(c4,c5)})
A_2 can be viewed (from the concept base viewpoint) as (CB_2, RC_2) =
({c2,c3,c6,c7},{r1(c2,c6),r1(c6,c7),r1(c3,c7)})
$overlp_{cl}(A_1, A_2) \Leftrightarrow R_2(A_1, A_2)$ and $CB_1 \cap CB_2 = \{c2, c3\}$, $RC_1 \cap RC_2 = \Phi$

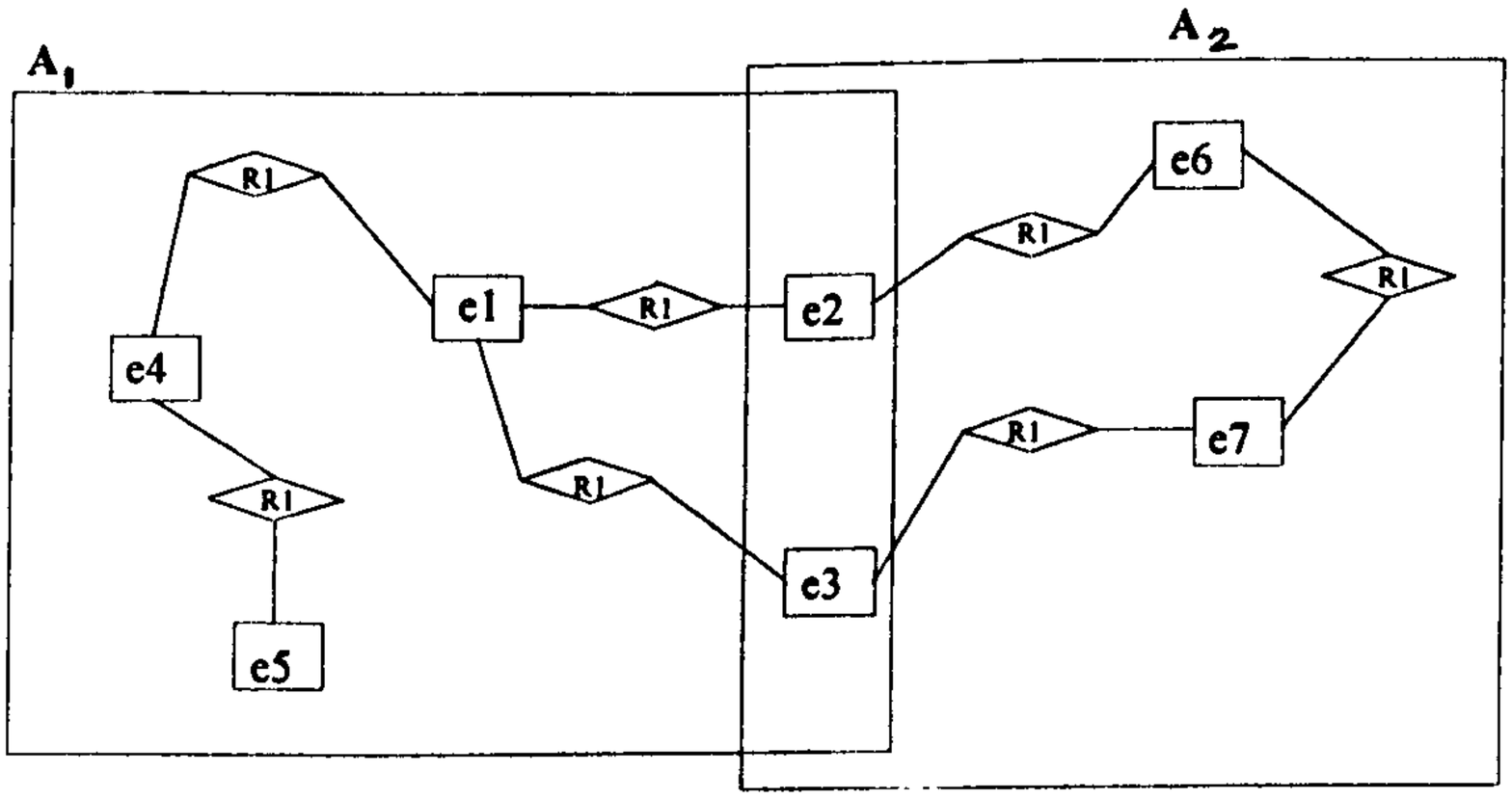


Figure 4.21 : An example of the concept level of the overlapped relation

The definition of the concept-relation level of the overlapped relation:

$$\text{overlp}_{\text{c-rl}}(A_{sci}, A_{scj}) \Leftrightarrow R_2(A_{sci}, A_{scj}) \text{ and } CB_{sci} \cap CB_{scj} \neq \Phi,\ RC_{sci} \cap RC_{scj} \neq \Phi$$

According to the above definition, an example of this type of overlapped relation, as shown in Figure 4.22, can be represented as follows.

A_1 can be viewed as (CB_1, RC_1) =

({c1,c2,c3,c4},{r1(c1,c2),r1(c1,c4),r1(c2,c3),r1(c3,c4)})

A_2 can be viewed as (CB_2, RC_2) =

({c1,c2,c3,c5,c6},{r1(c1,c2),r1(c2,c3),r1(c2,c5),r1(c2,c6)})

$\text{overlp}_{\text{c-rl}}(A_1, A_2) \Leftrightarrow R_2(A_1, A_2)$ and

$CB_1 \cap CB_2 = \{c1,c2,c3\},\ RC_1 \cap RC_2 = \{r1(c1,c2),r1(c2,c3)\}$

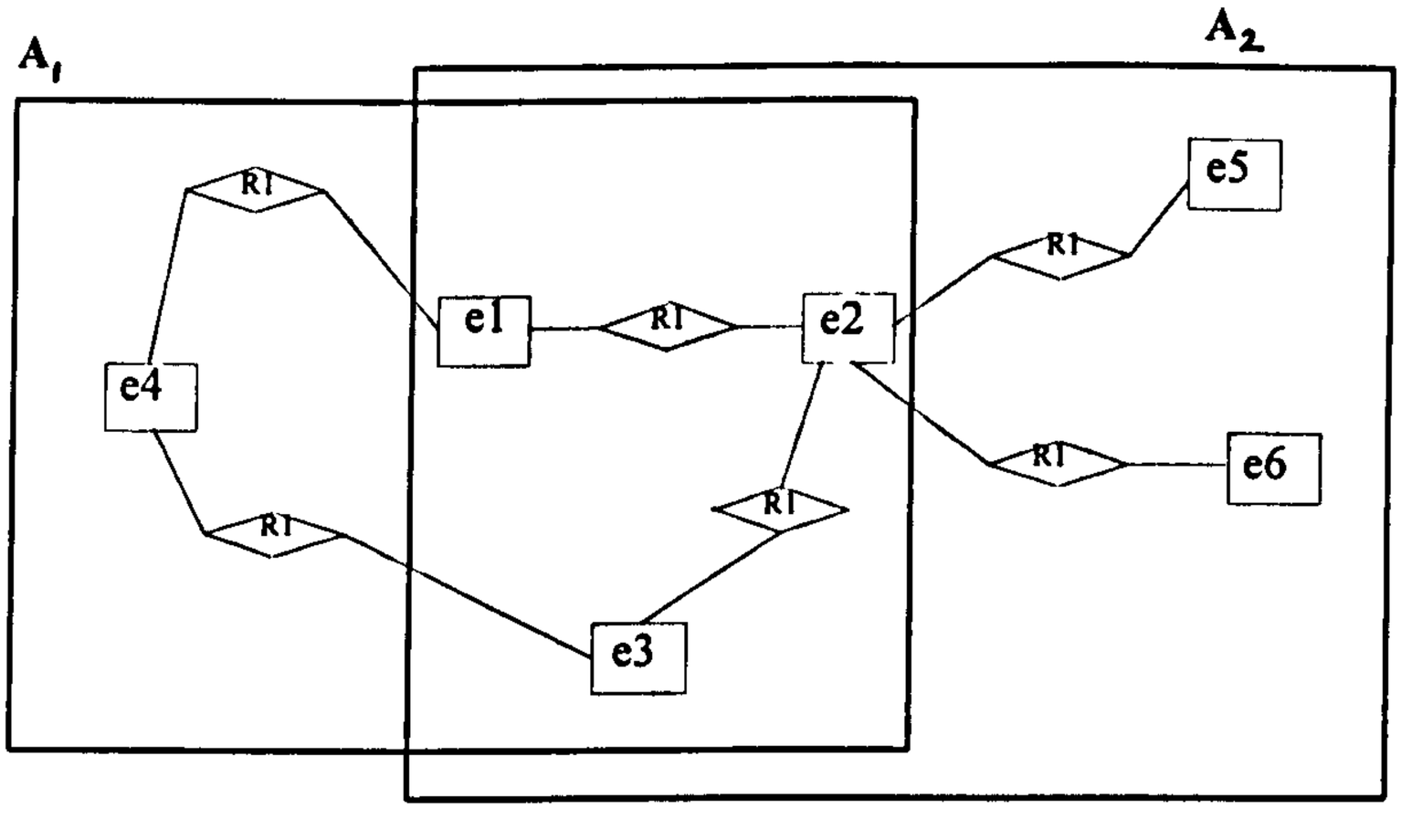


Figure 4.22 : An example of the concept-relation level of the overlapped relation

4.4.5 The Formalisation of the Case Base

Based on the above definitions of the case and the relations among cases, the case base can be formalised as follows.

Definition: C is the case base. Let C be defined as a triple of sets as follows.

$$C = (AC, RA, SE)$$

where $AC = \bigcup_{i=1,\ldots,m} \bigcup_{j_i} ASC_{j_i,i} \cup A_{dci}$

$$= (\bigcup_{j_1=1,\ldots,n_1} ASC_{j_1,1} \cup A_{dc1}) \cup (\bigcup_{j_2=1,\ldots,n_2} ASC_{j_2,2} \cup A_{dc2}) \cup \ldots \cup (\bigcup_{j_m=1,\ldots,n_m} ASC_{j_m,m} \cup A_{dcm})$$

$$= (ASC_{1,1} \cup ASC_{2,1} \cup \ldots \cup ASC_{n_1,1} \cup A_{dc1}) \cup (ASC_{1,2} \cup ASC_{2,2} \cup \ldots \cup ASC_{n_2,2} \cup A_{dc2}) \cup \ldots \cup (ASC_{1,m} \cup ASC_{2,m} \cup \ldots \cup ASC_{n_m,m} \cup A_{dcm})$$

A_{dci}: the domain case in the i domain

$ASC_{j_i,i}$: a set of cases of the j_i system in the i domain, i.e.

$$= \{A_{sc1,j_i,i}, A_{sc2,j_i,i}, \ldots, A_{sck_{j_i},j_i,i}\}$$

$$RA = (\bigcup_i R_1(A_{scai}, A_{scbi})) \cup (\bigcup_i R_2(A_{scxi}, A_{scyi})) \cup (\bigcup_i \bigcup_{j_i} R_3(A_{t-sc_{j_i},j_i,i}, A_{dci})),$$

including only three types of relation of cases,

$R_1(A_{scai}, A_{scbi})$: the proper subset-superset relation of A_{scai} and A_{scbi} system cases, i.e. $A_{scai} \subset A_{scbi}$

$R_2(A_{scxi}, A_{scyi})$: the overlapped relation of A_{scxi} and A_{scyi} system cases

$R_3(A_{t-sc_{j_i},j_i,i}, A_{dci})$: the is-typical link of a domain case and its typical case of j_i system in the i domain

$SE = \{e_{s1}, e_{s2}, \ldots\}$ is a set of non-case (stand alone) entities in the case base.

According to the above definition, an example of the case base, as shown in Figure 4.23, can be written as follows.

$$AC = \bigcup_{i=1,\ldots,m} \bigcup_{j_i} ASC_{j_i,i} \cup A_{dci} \cup SE$$

(because of only one domain and two kinds of system, i.e. $m = 1$, $j_i = 1,2$)

$$= \bigcup_{j_1=1,2} ASC_{j_1,1} \cup A_{dc1}$$

$$= ASC_{1,1} \cup ASC_{2,1} \cup A_{dc1}$$

$$= \{A1,A3,A5,A12,A15,A17\} \cup \{A20,A23,A25,A28,A30) \cup \{ADCX\}$$

$$= \{A1,A3,A5,A12,A15,A17,A20,A23,A25,A28,A30,ADCX\}$$

$$RA = (\bigcup_{i} R_1(A_{scai}, A_{scbi})) \cup (\bigcup_{i} R_2(A_{scxi}, A_{scyi})) \cup (\bigcup_{i} \bigcup_{j_1} R_3(A_{t-sc_{ki},j_1,i}, A_{dci}))$$

$$= (\{R_1(A5,A3), R_1(A3,A12), R_1(A1,A17), R_1(A17,A15), R_1(A25,A20),$$
$$R_1(A20,A30), R_1(A23,A28)\}) \cup$$
$$(\{R_2(A5,A17), R_2(A5,A15), R_2(A3,A17), R_2(A3,A15), R_2(A12,A17),$$
$$R_2(A12,A15), R_2(A15,A30), R_2(A20,A28), R_2(A28,A30)\} \cup$$
$$(\{R_3(A3,ADCX), R_3(A20,ADCX)\})$$

$$= \{R_1(A5,A3), R_1(A3,A12), R_1(A1,A17), R_1(A17,A15), R_1(A25,A20),$$
$$R_1(A20,A30), R_1(A23,A28), R_2(A5,A17), R_2(A5,A15), R_2(A3,A17),$$
$$R_2(A3,A15), R_2(A12,A17), R_2(A12,A15), R_2(A15,A30), R_2(A20,A28),$$
$$R_2(A28,A30), R_3(A3,ADCX), R_3(A20,ADCX)\}$$

$SE = \Phi$

Case base = (AC,RA,SE) =

$$(\{A1,A3,A5,A12,A15,A17,A20,A23,A25,A28,A30,ADCX\},$$
$$\{R_1(A5,A3), R_1(A3,A12), R_1(A1,A17), R_1(A17,A15), R_1(A25,A20),$$
$$R_1(A20,A30), R_1(A23,A28), R_2(A5,A17), R_2(A5,A15), R_2(A3,A17),$$
$$R_2(A3,A15), R_2(A12,A17), R_2(A12,A15), R_2(A15,A30), R_2(A20,A28),$$
$$R_2(A28,A30), R_3(A3,ADCX), R_3(A20,ADCX)\})$$

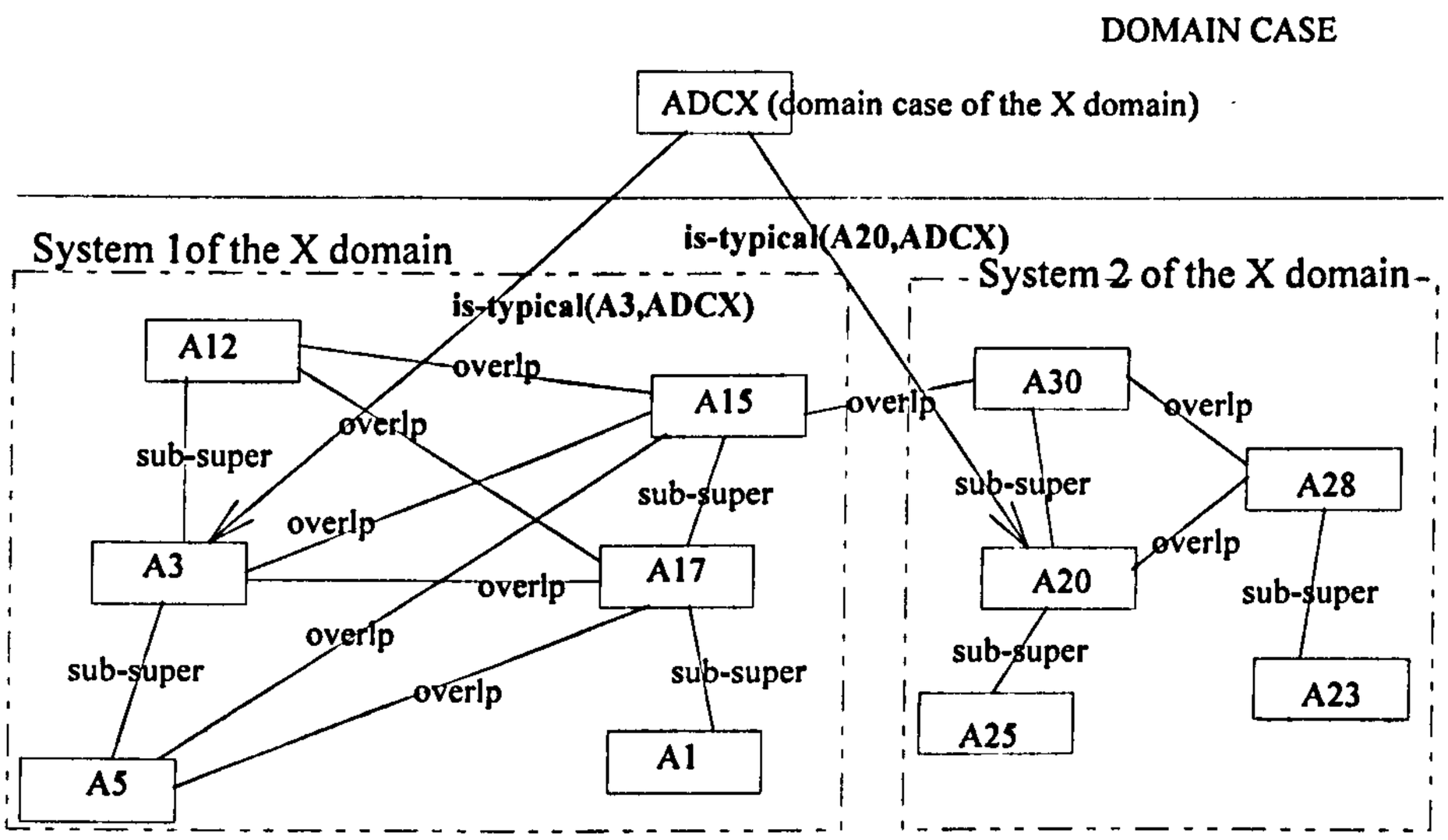


Figure 4.23 : An example of the case base

4.5 The Intrinsic Properties of the Relations in the DBKM for Retrieving the Relevant Concrete Experiences

By reusing similar past cases, designers can draw a whole picture of the problem domain for comprehending the problem, recognising certain requirements, and eliciting other relevant information. By reusing similar past cases, designers can pre-understand the problem domain and ease the communication with users, and the difficulties facing the designers - the problems of object identification and interpretation - can be facilitated. Therefore, accessing similar cases is the primary issue in the DBKM environment. In this section, illustrations are given first of the inherent properties of the superset, overlapped and proper subset relations among system cases, secondly of the exemplar-link relation(s) of a concept and its

corresponding entity(-ies), and lastly of the typical-link relations of a domain case and its typical system cases. By means of these inherent properties, not only whole similar cases can be efficiently retrieved, but also the relevant parts of the conceptual schemata in different cases can be accessed from the very large and complicated concrete experiences base (case base).

4.5.1 The Internal General Structures of a Concept and System Case

Before illustrating the inherent properties of the relations in the DBKM for retrieving the relevant concrete experiences, the internal general representation of a concept in the concept base, including its exemplars in the case base, and the internal general representation of a case structure, including its corresponding part of the concept base, will be exemplified.

Based on the formalisations defined in sections 4.4.1 and 4.4.3, the general representation of a concept, such as the "a" concept in the concept base and its exemplars in the case base, as shown in Figure 4.24, are presented as follows:

(1) Exemplar links: has-exemplar(a, A), has-exemplar(a, A'), has-exemplar(a, A'').

(2) Relations

the non-directional undefined relation(U_D): U_D(a,b), U_D(a,c).

the directional IS-A relation: IS-A(w,a), IS-A(y,a), IS-A(z,a).

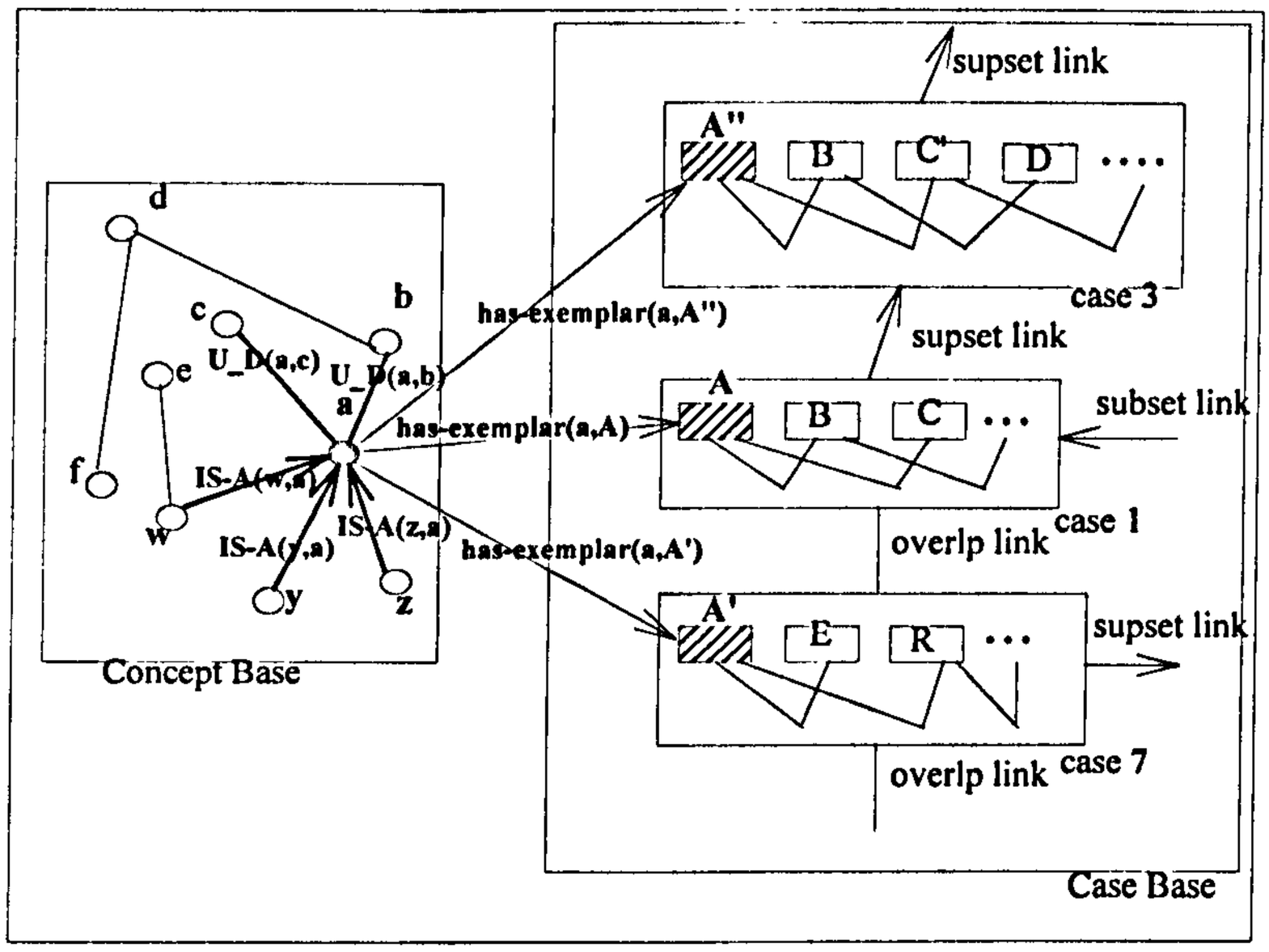


Figure 4.24 : The general representation of a concept in the concept base

The general representation of a case and its corresponding part in the concept base, as shown in Figure 4.25, are illustrated as follows.

The structures of the entities and relationships between the related entities in a case are represented as follows.

(1) The structure of an entity (using A entity as an example):

E(A; attributes) means A is an entity and its attributes.

(2) The structure of the relationship between related entities (using the relationship of entities A and B as an example):

$R1_1$(A(1,1),B(1,n); nil) means the relationship between A and B; the cardinality of the A side is (1,1) and the cardinality of the B side is (1,n); there are no attributes in the relationship of A and B.

Based on the formalisations defined in sections 4.4.3, the one-to-one correspondence of a case 1 and its corresponding elements in the case base are presented as follows.

(3) one-to-one correspondence: (using case 1 as an example)

A - a, B - b, C - c, D - d, $R1_1$(A,B) - r1(a,b), $R1_1$(B,C) - r1(b,c), $R1_1$(A,D) -r1(a,d)

Based on the formalisations of the two relations among system cases discussed in section 4.4.4, the structures of the three links in a case are represented as follows.

(3) The structure of the supset link:

added(case-number; concept(s), relation(s))

For example, the structure of the supset link of case 1 is added(case 3; concept(s), relation(s))

(4) The structure of the subset link:

deleted(concept(s), relation(s); case-number)

For example, the structure of the subset link of case 1 is deleted(concept(s), relation(s); case 5)

(5) The structure of the overlapped link:

overlp(case number; the type of overlapped relation; the overlapped degree; the difference of the overlapped case)

where

1. The type of the overlapped relation: the concept level or the concept-relation level.
2. The overlapped degree: using the equation in section 4.3.3.1. .
3. The difference of the overlapped case (derived from the notion of the difference link used in PROTOS as discussed in section 3.3.1): It is used to check whether the overlapped case included the concerned entity (-ies). The difference of the overlapped case is from the viewpoint of the concept base, then the contents of the difference are the concepts (relations) which correspond to the different entities (relationships) in the overlapped case.

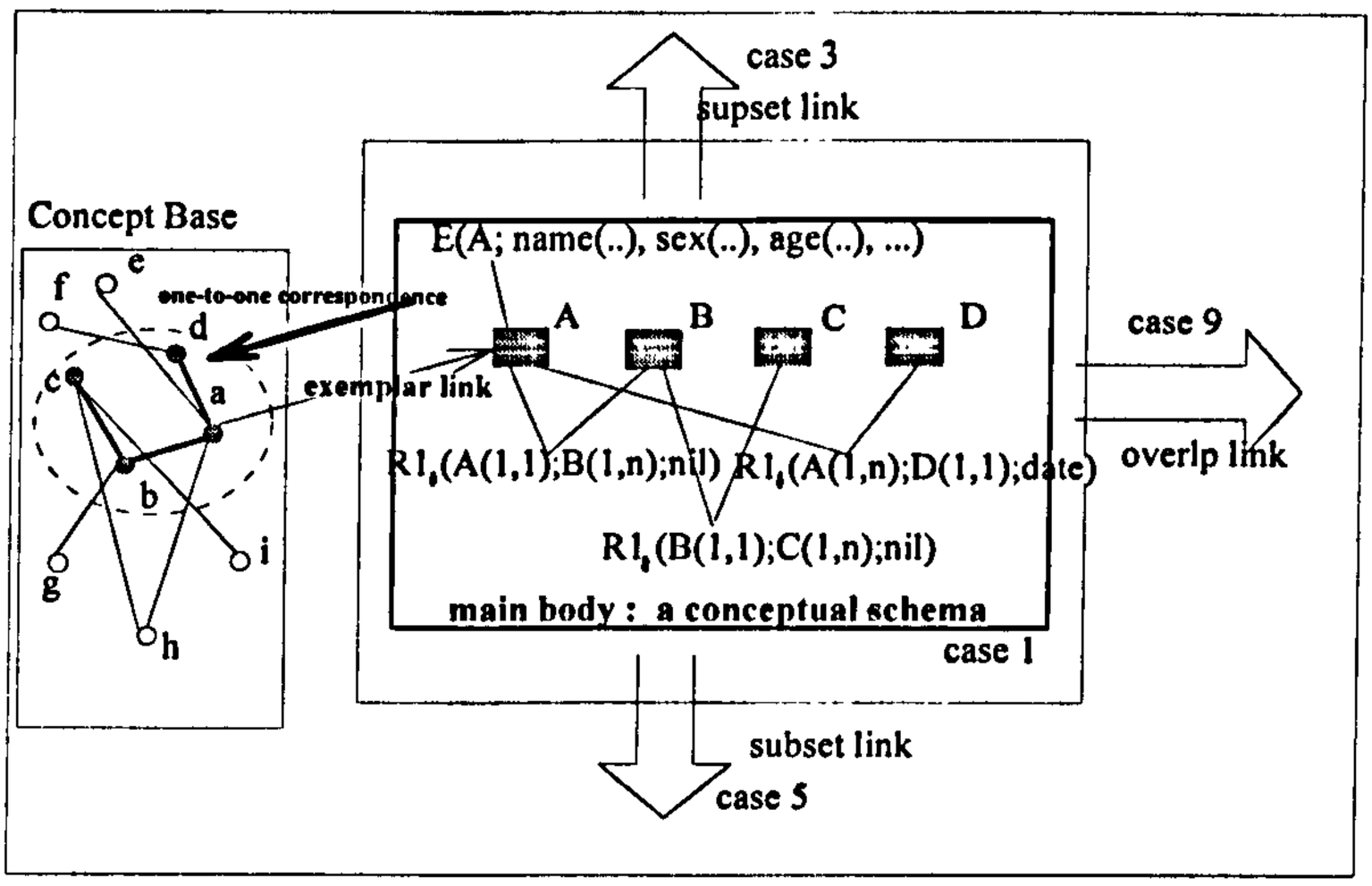


Figure 4.25 : The general representation of a case structure and the corresponding part in the concept base

4.5.2 The Property of the Relations between System Cases

In order to access similar system cases from the case base, the three types of link - the supset, overlapped and subset links discussed above - are used to construct the index structure of system cases. In this section, the inherent properties of these relations which can be applied efficiently to retrieve the relevant system cases from the very large case base are illustrated in turn.

4.5.2.1 The property of the superset relation (supset link)

For the purpose of clarifying the inherent properties of the superset relations and thus facilitating the retrieval of relevant system cases, the derived properties from the finding element and from the partial ordering relation (the superset relation) are exemplified as follows.

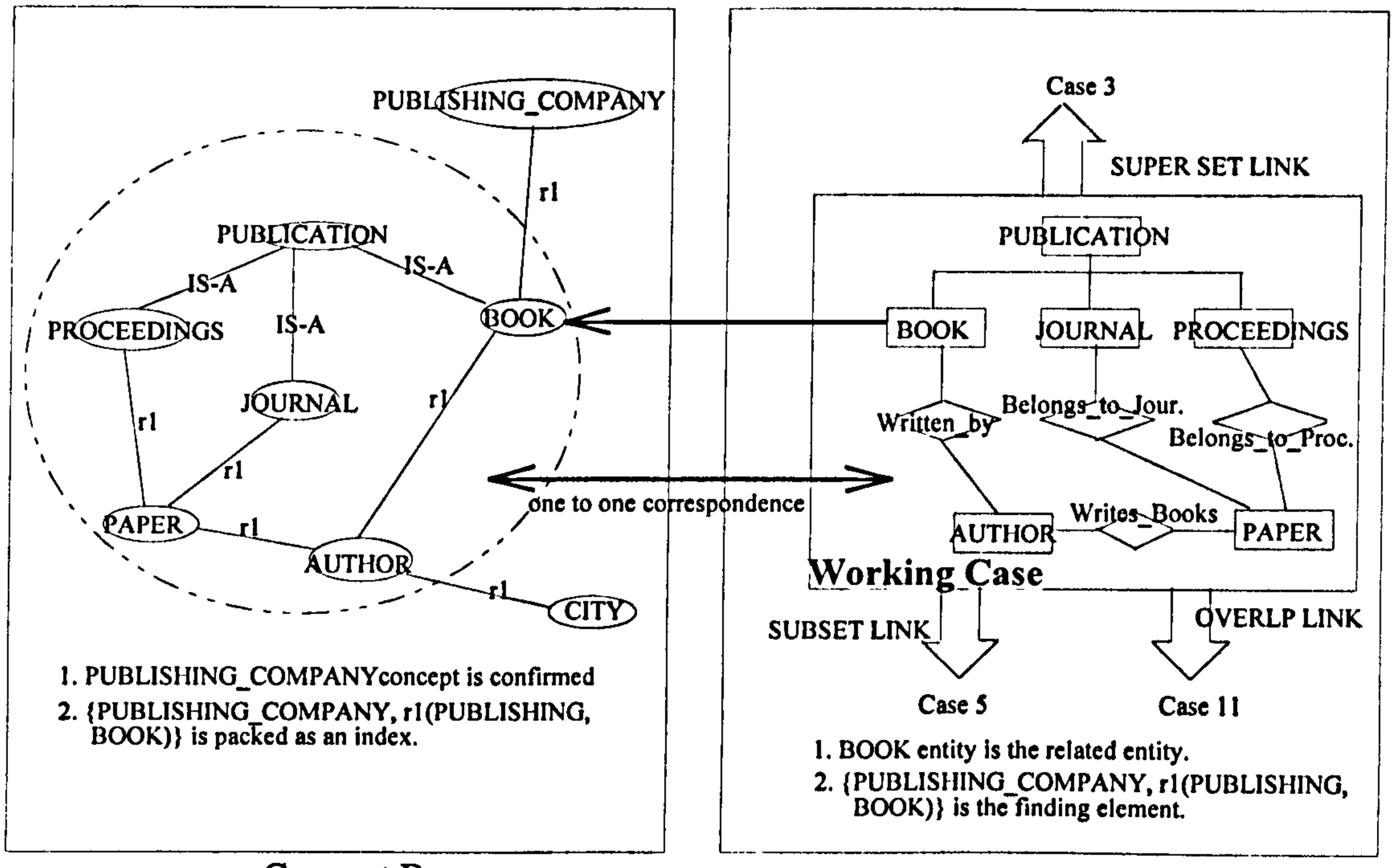
(I) The derived property from the finding element

(1) Definitions

Basically, the finding element is a set which consists of the confirmed concept(s) and its (their) relation(s) with the related concept(s). For instance, the *working case*, which is a case that is used as a reference point to retrieve further relevant cases, includes six entities: BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER entities. The six corresponding concepts - BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER - and the relations among them in the concept base are shown in Figure 4.26. If the PUBLISHING_COMPANY is the confirmed concept, then the finding element is {PUBLISHING_COMPANY, r1(PUBLISHING_COMPANY, BOOK)}[52] , which will be used as a clue for retrieving relevant cases involving the corresponding entity (relationship) from the working case.

---

[52] Basically the confirmed concept and the relation with the related concept will be packed as an index set. The index set will be partitioned into two or more subsets if there are either two more objects or two more relations in the index pack. The maximum finding element is the index set itself. In this example, because there is only one confirmed concept, there is only one finding element, which consists of the confirmed concept and its relation with the related concept such as BOOK in this example, and is also the maximum finding element. A more complicated situation, for example with more than one confirmed concept, will be discussed in chapter 5.



(1) Working Case : { BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, IS-A(BOOK,PUBLICATION), IS-A(PROCEEDINGS,PUBLICATION), IS-A(JOURNAL,PUBLICATION), Written_by(PAPER,AUTHOR), Writes_Books(BOOKS,AUTHOR), Belongs_to_Jour.(PAPER,JOURNAL), Belongs_to_Proc.(PAPER,PROCEEDINGS)}.

(2) The corresponding part of the working case in the concept case : { BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, IS-A(BOOK,PUBLICATION), IS-A(PROCEEDINGS,PUBLICATION), IS-A(JOURNAL,PUBLICATION), r1(PAPER,AUTHOR), r1(BOOKS,AUTHOR), r1(PAPER,JOURNAL), r1(PAPER,PROCEEDINGS)}.

(3) The elements in the working case and the corresponding elements in the concept base are in one-to-one correspondence.

(4) The PUBLISHING_COMPANY concept is the confirmed concept (the related concept is the BOOK concept).

(5) According to the discussion in section 4.3.2, the index set {PUBLISHING_COMPANY,r1(PUBLISHING_COMPANY,BOOK)}will be transferred into the case base.

(6) In this example, the finding element is the same as the index set: {PUBLISHING_COMPANY,r1(PUBLISHING_COMPANY,BOOK)}.

Figure 4.26 : An example of the dual-base knowledge structure

(2) The existing property for the finding element

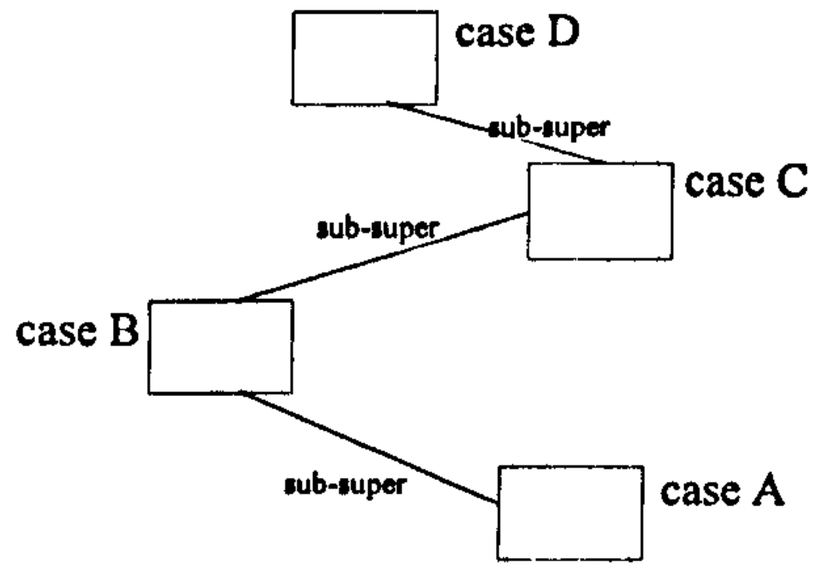
Based on the duality of the concept and case bases, the corresponding entity (relationship) of the finding element must be included in some cases in the case base. Based on the definition of the two relations defined in section 4.2.2.2, the relevant case(s), including the finding element, can be found by means of the relations of the working case. (Because the working case includes the related corresponding entity, BOOK, in this example, the case(s) including the finding element must be related with the working case). In other words, the relevant case(s) involving the finding element is (are) either in the superset case(s) or in the overlapped case(s) of the working case or both.

(II) The derived properties from the partial ordering relation (the superset relation)

Basically, the process of retrieving the relevant super-case(s) of the working case involves travelling up from the working case until the end. The circumstance encountered here is when the maximum finding element is included in many super-case(s) which is(are) not the final super-case(s) of the working case. The situation can be tackled by the intrinsic property of the superset relation as illustrated below.

(1) Definitions

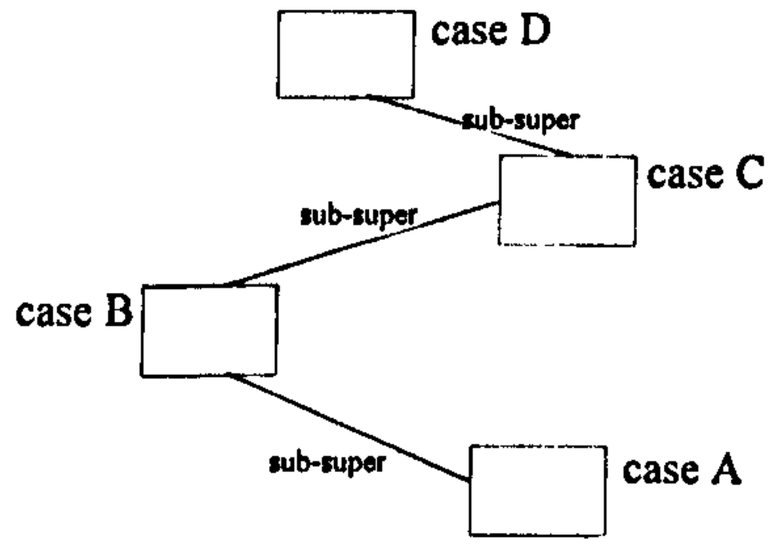
Definition: The *super-case (s)* of one case is (are) the case(s) which is (are) of the proper super set of the case. For example, as shown in Figure 4.27, if A R_1 B, B R_1 C, C R_1 D, then cases B, C, D are the super-cases of A; cases C, D are the super-cases of B; case D is the super-case of C.



the super-cases of A: B, C, D
the super-cases of B: C, D
the super-case of C: D

Figure 4.27 : An example of the super-case

Definition: The *sub-case(s)* of one case is (are) the case(s) which is (are) of the proper subset of the case. For example, as shown in Figure 4.28, if A R_1 B, B R_1 C, C R_1 D, then cases C, B, A are the sub-cases of A; cases B, A are the sub-cases of C; case A is the sub-case of C.

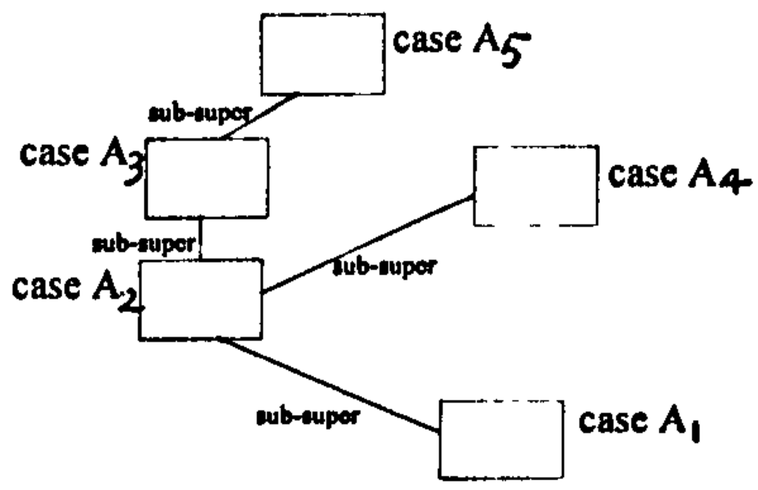


the sub-cases of D: C, B, A
the sub-cases of C: B. A
the sub-case of B: A

Figure 4.28 : An example of the sub-case

Definition: Let A = $\{A_i \mid i \in N\}$ and the elements of A are cases. A is a strict partial order set, i.e. the cases in A exist with proper subset-superset relations. Case A_i is called a *relative maximal case* of A if none of the cases of A strictly includes A_i, i.e. A_i has no any super-case. For example, as shown in Figure 4.29, if A =

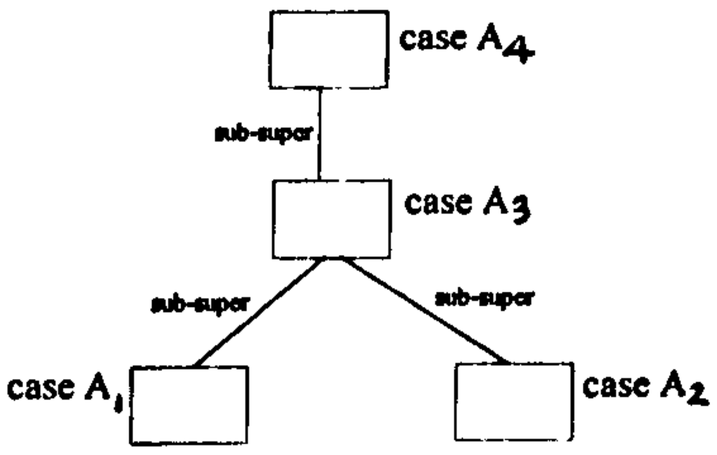
$\{A_1, A_2, A_3, A_4, A_5\}$and $A_1 R_1 A_2$, $A_2 R_1 A_3$, $A_2 R_1 A_4$, $A_3 R_1 A_5$, then cases A_4, A_5 are the relative maximal cases of A.



A = $\{A_1, A_2, A_3, A_4, A_5\}$
The relative maximal cases of A are A_4 and A_5

Figure 4.29 : An example of the relative maximal case

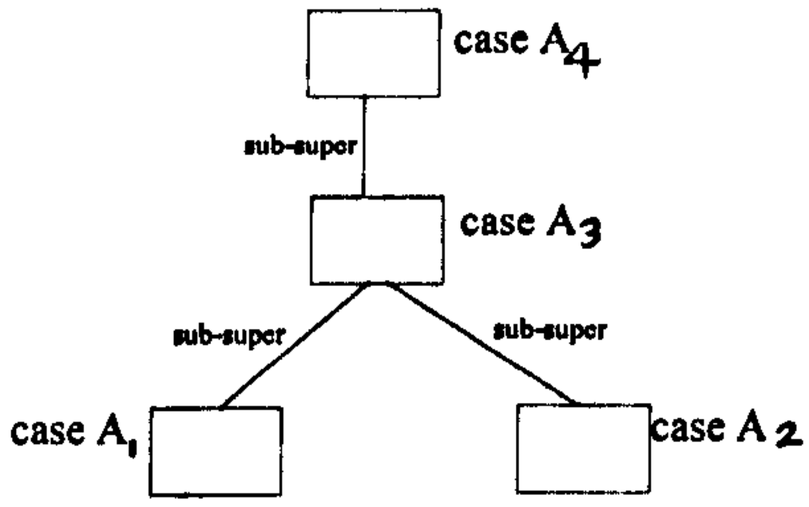
Definition: Let A = $\{A_i \mid i \in N\}$ and the elements of A are cases. A is a strict partial order set, i.e. the cases in A exist with proper subset-superset relations. Case A_i is called a *relative minimal case* of A if none of the cases of A is strictly included by A_i, i.e. A_i has no sub-case. For example, as shown in Figure 4.30, if A = $\{A_1, A_2, A_3, A_4\}$and $A_1 R_1 A_3$, $A_2 R_1 A_3$, $A_3 R_1 A_4$, then cases A_1, A_2 are the relative minimal cases of A.



A = $\{A_1, A_2, A_3, A_4\}$
The relative minimal cases of A are A_1 and A_2

Figure 4.30: An example of the relative minimal case

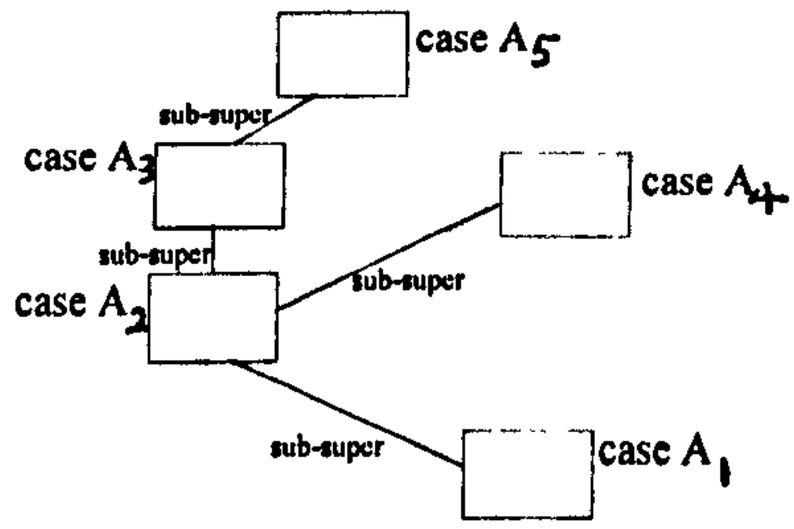
Definition: Let A = $\{A_i \mid i \in N\}$ and the elements of A are cases. A is a strict partial order set, i.e. the cases in A exist with proper subset-superset relations. Case A_i is called a *greatest case* of A if $A_j \subset A_i$ for every $A_j \in A$, $i \neq j$, i.e. the greatest case of A is unique. For example, as shown in Figure 4.31, if A = $\{A_1, A_2, A_3, A_4\}$ and $A_1 R_1 A_3$, $A_2 R_1 A_3$, $A_3 R_1 A_4$, then case A_4 is the greatest case of A.



A = $\{A_1, A_2, A_3, A_4\}$
The greatest case of A is A_4

Figure 4.31 : An example of the greatest case

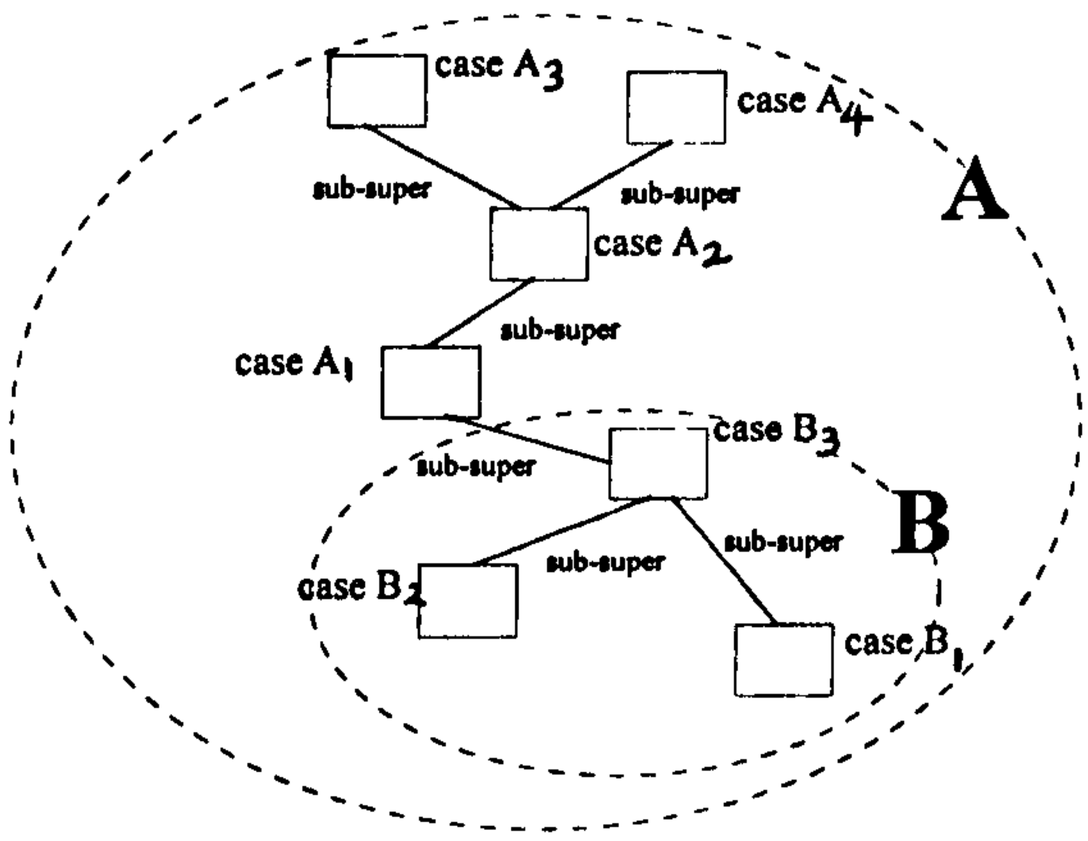
Definition: Let A = $\{A_i \mid i \in N\}$ and the elements of A are cases. A is a strict partial order set, i.e. the cases in A exist with proper subset-superset relations. Case A_i is called a *least case* of A if $A_i \subset A_j$ for every $A_j \in A$, $i \neq j$, i.e. the least case of A is unique. For example, as shown in Figure 4.32, if A = $\{A_1, A_2, A_3, A_4, A_5\}$ and $A_1 R_1 A_2$, $A_2 R_1 A_3$, $A_2 R_1 A_4$, $A_3 R_1 A_5$, then case A_1 is the least case of A.



A = $\{A_1, A_2, A_3, A_4, A_5\}$
The least case of A is A_1

Figure 4.32 : An example of the least case

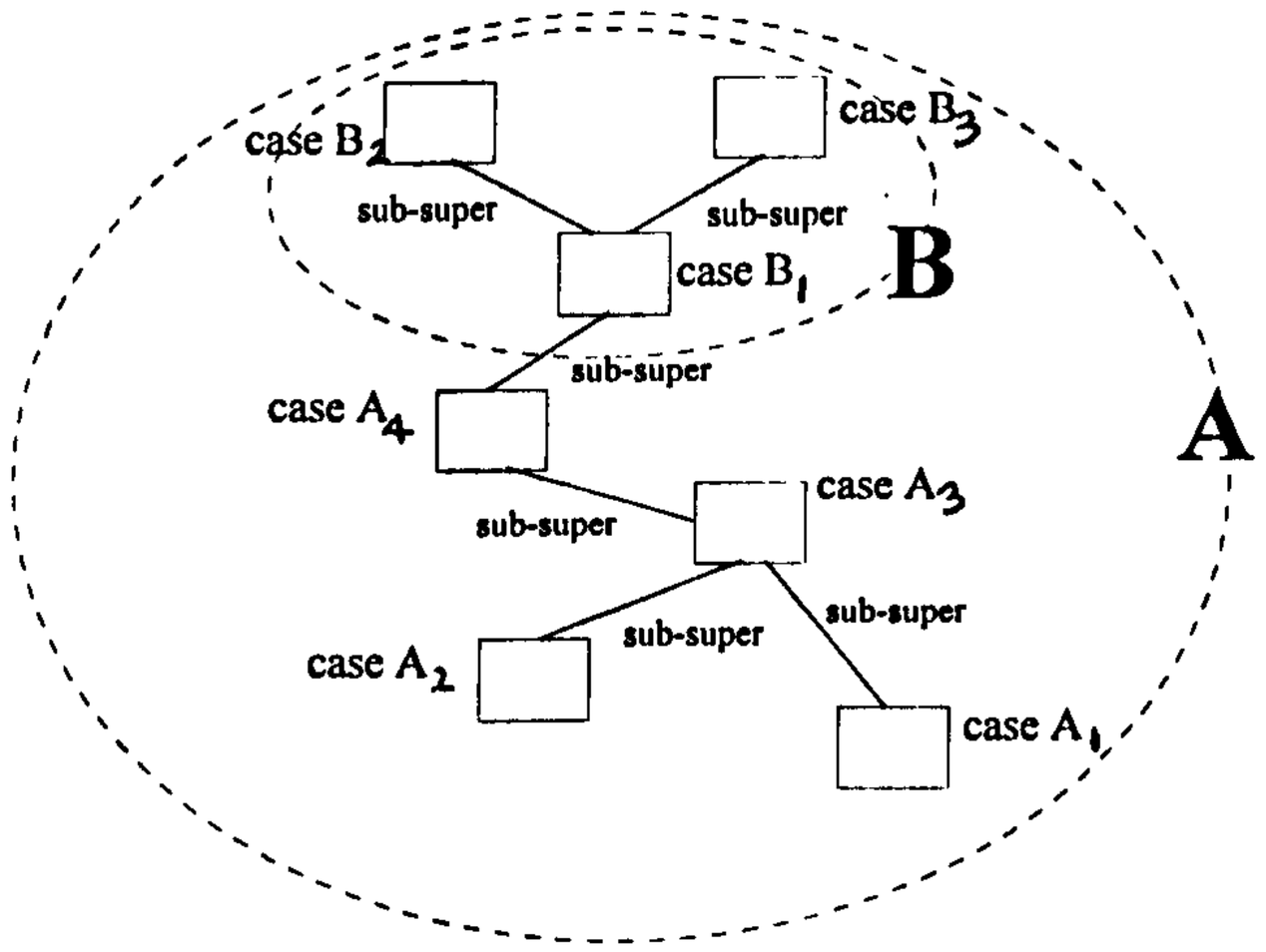
Definition: Let B = $\{B_i \mid i \in N\}$ be a proper subset of A = $\{A_i \mid i \in N\}$ and not include the finding element. A, B are a strict partial order set, i.e. the cases in A and B exist with proper subset-superset relations. *An upper bound of B in A including the finding element* is a case $A_i \in A$ such that $B_i \subset A_i$ for every B_i, as shown in Figure 4.33.



A = $\{A_1, A_2, A_3, A_4, B_1, B_2, B_3\}$ and cases
A_1, A_2, A_3, A_4 include the finding element.
B = $\{B_1, B_2, B_3\}$ does not include the finding element.
The upper bounds of B in A, including the finding element, are
A_1, A_2, A_3, A_4 cases.

Figure 4.33 : An example of the upper bound

Definition: Let B = $\{B_i \mid i \in N\}$ be a proper subset of A = $\{A_i \mid i \in N\}$ and include the finding element. A, B are a strict partial order set, i.e. the cases in A and B exist with proper subset-superset relations. *A lower bound of B in A that does not include the finding element* is a case $A_i \in A$ such that $A_i \subset B_i$ for every B_i, as shown in Figure 4.34.



A = $\{A_1, A_2, A_3, A_4, B_1, B_2, B_3\}$ and cases
A_1, A_2, A_3, A_4 do not include the finding element.
B = $\{B_1, B_2, B_3\}$ includes the finding element.
The lower bounds of B in A that do not include the finding element are A_1, A_2, A_3, A_4 cases.

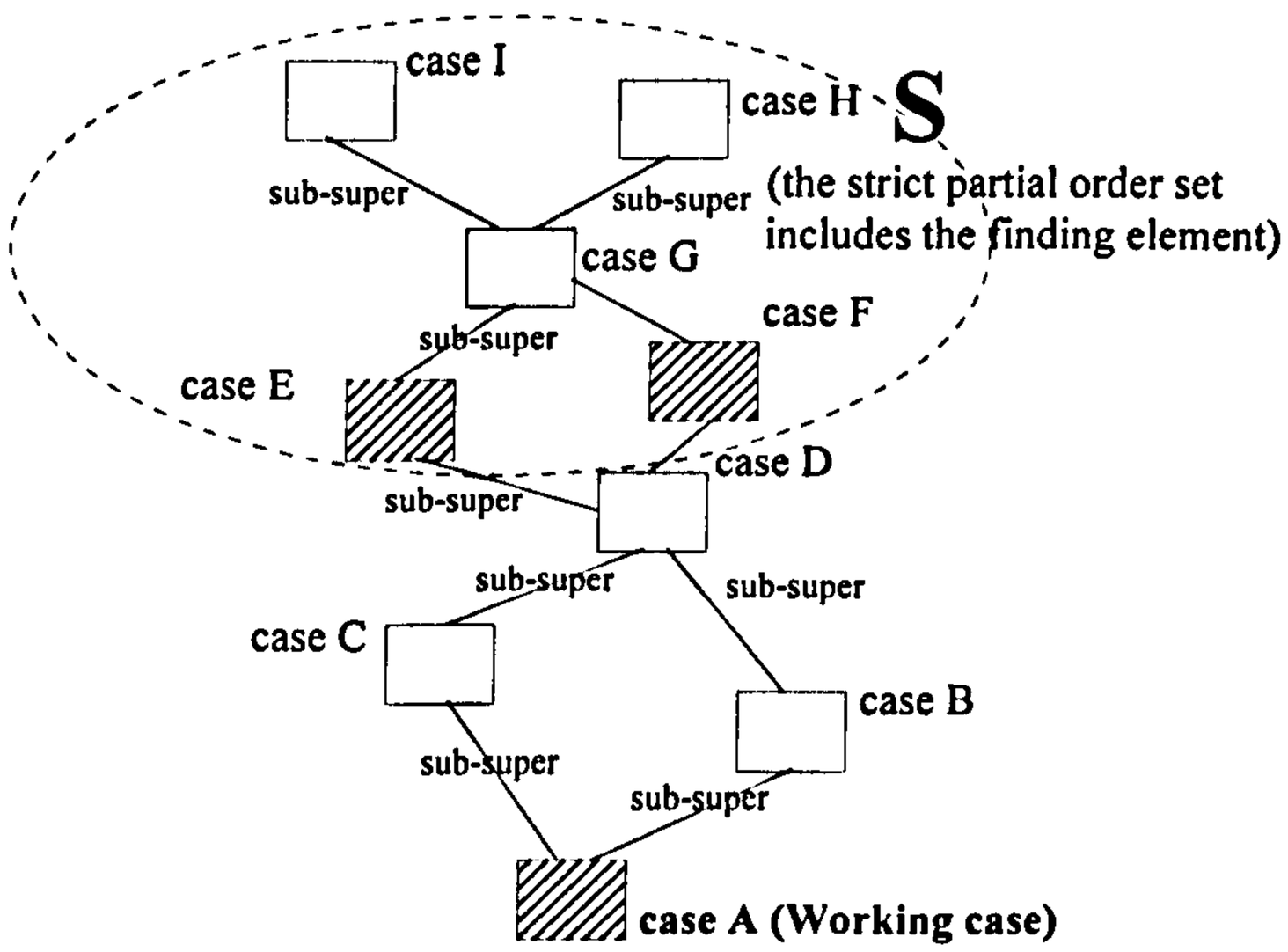
Figure 4.34 : An example of the lower bound

Definition: If the set of upper bounds of B in A has a least case, then the case is called the *least upper bound* of B in A. The least upper bound of B in A is also called the *supremum* of B in A, abbreviated $\sup_A B$. According to the previous example (the upper bound), the $\sup_A B$ is case A_1.

Definition: If the set of lower bounds of B in A has a greatest case, then the case is called the *greatest lower bound* of B in A. The greatest lower bound of B in A is also called the *infimum* of B in A, abbreviated $\inf_A B$. According to the previous example (the lower bound), the $\inf_A B$ is case A_4.

(2) The properties for exiting the retrieval process

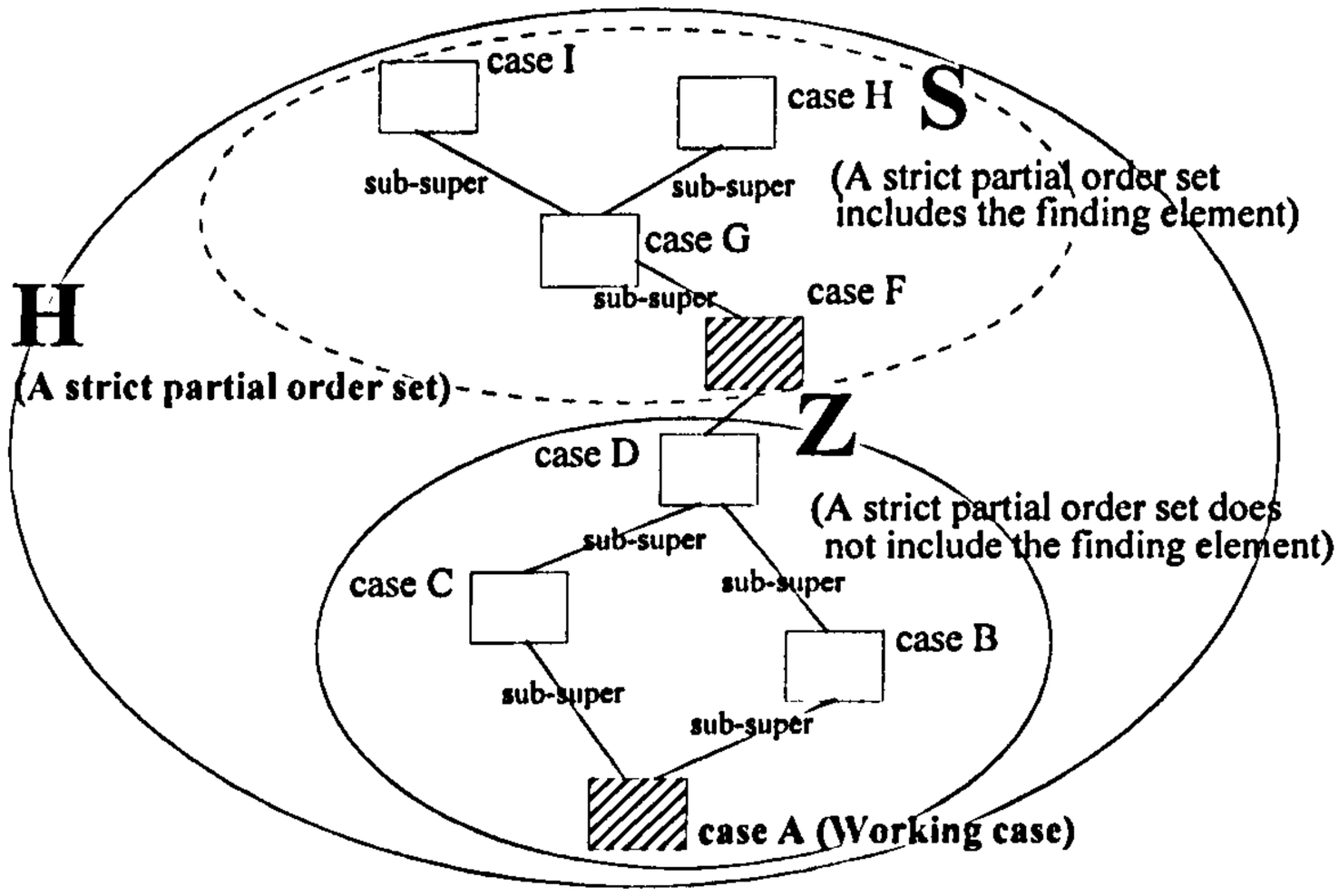
Based on the above definitions, if the maximum finding element is in the super-cases, but not in the final super-cases of the working case, there must exist at least one relative minimal case of the strict partial order set (there might be more than one kind of this set), which consists of the super-cases, including the maximum finding element of the working case, as shown in Figure 4.35.



The relative minimal cases of S are case E and case F.

Figure 4.35 : An example of the relative minimal case of a partial order set

If the strict partial order set as defined above, such as the S set in Figure 4.35, exists in only one relative minimal case, there is a supremum of the set, which is a strict partial order set consisting of the sub-cases of this relative minimal case which do not include the finding element, such as the Z set in Figure 4.36.



The least upper bound of Z in H is case F (i.e. sup_H = case F)

Figure 4.36 : An example of the supremum

Using these two properties, when the first encountered super-case includes the finding element of the working case, the process of travelling up from the working case by the supset links can be terminated.

4.5.2.2 The property of the overlapped relation (overlapped link)

Two useful properties of the overlapped relation can be used together to handle the process of retrieving the relevant overlapped cases. The first one is the type of the overlapped relation and the second is the overlapped degree. As discussed in section 4.2.2.2 and 4.4.4, there are two types of the overlapped relation: the concept level and the concept-relation level. A reasonable way to think about using these types of overlapped relation in the process of system case retrieval is to suggest that accessing the overlapped cases in the concept-relation level is more helpful for accessing the

overlapped cases in the concept level. Therefore, when the overlapped links are used to find out whether the overlapped cases of the working case include the finding element, the overlapped cases of the concept-relation level take priority over those in the concept level. The overlapped degree can be used control the number of the retrieved similar overlapped cases.

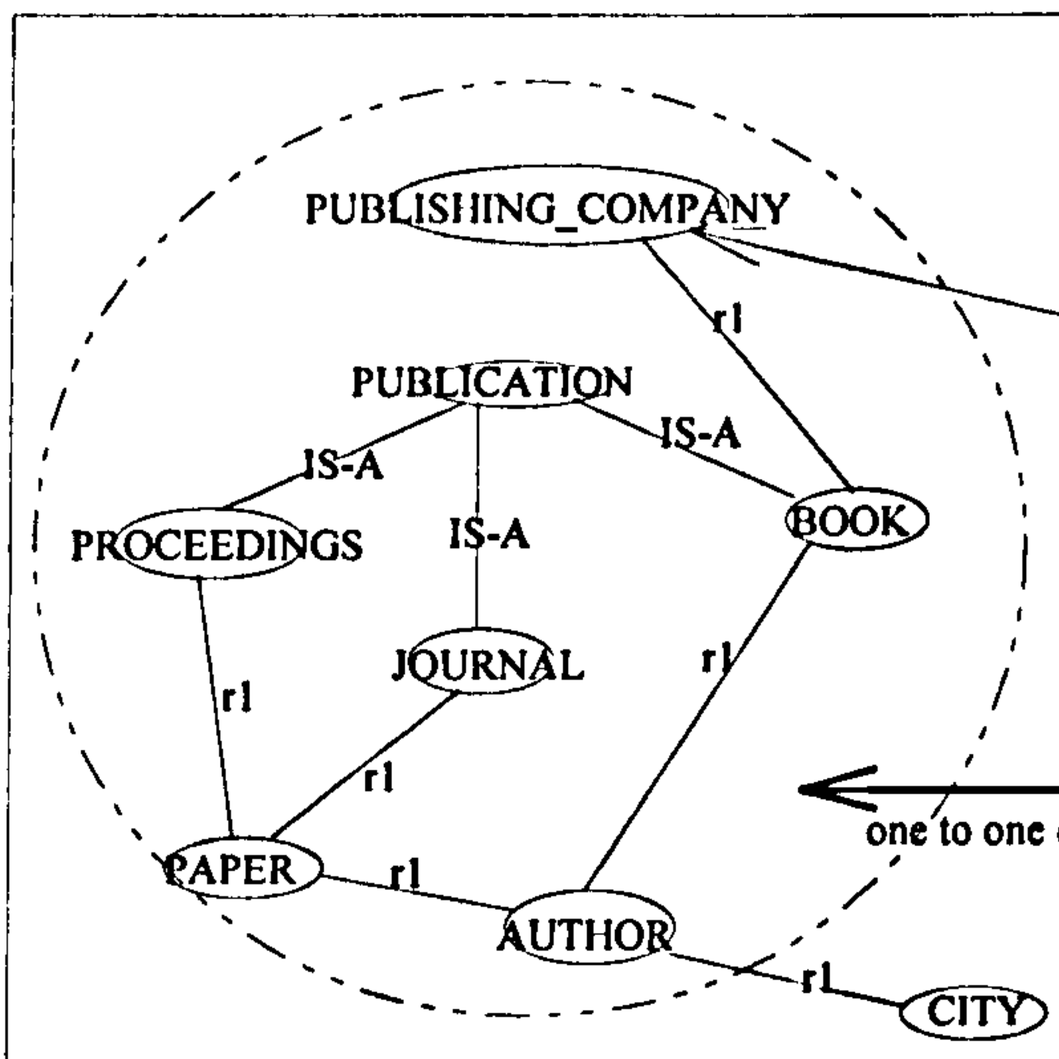
4.5.2.3 The property of the proper subset relation (subset link)

For clarifying the inherent properties of the subset relations in helping the retrieval relevant system cases, the notion of the deleting element and the derived property from the subset relation to terminate the retrieval process are exemplified as follows.

(I) The deleting element

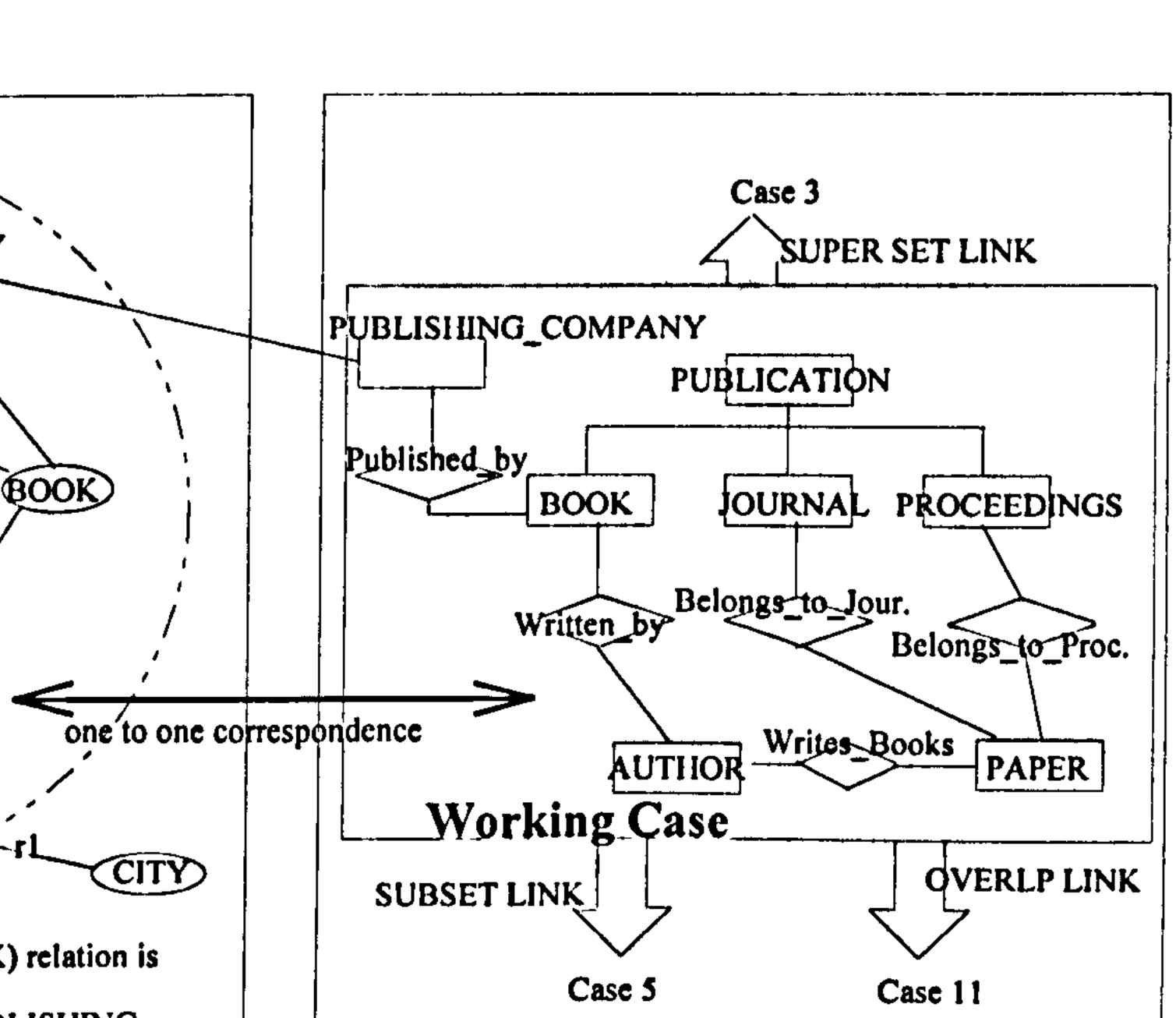
Basically, the deleting element is a set which consists of the deleting concept(s) and its(their) relation(s) with the related concept(s). For instance, the *working case* includes seven entities: BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, PUBLISHING_COMPANY entities. The six corresponding concepts - BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, PUBLISHING_COMPANY - and the relations among them in the concept base are shown in Figure 4.37. If the PUBLISHING_COMPANY is the deleted concept, then the deleting element is {PUBLISHING_COMPANY,

r1(PUBLISHING_COMPANY, BOOK)}[53], which will be used as a clue for retrieving relevant cases involving the non-deleted entity (relationship) from the working case.



1. R(PUBLISHING_COMPANY,BOOK) relation is confirmed.
2. {PUBLISHING_COMPANY, r1(PUBLISHING, BOOK)} is packed as an index.

Concept Base



1.PUBLISHING_COMPANY is the deleted entity.
2. {PUBLISHING_COMPANY, r1(PUBLISHING, BOOK)} is the deleting element.

Case Base

one to one correspondence

(1) Working Case : { BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, PUBLISHING_COMPANY, IS-A(BOOK,PUBLICATION), IS-A(PROCEEDINGS,PUBLICATION), IS-A(JOURNAL,PUBLICATION), Written_by(PAPER,AUTHOR), Writes_Books(BOOKS,AUTHOR), Belongs_to_Jour.(PAPER,JOURNAL), Belongs_to_Proc.(PAPER,PROCEEDINGS), Published_by(BOOK,PUBLISHING_COMPANY)}.
(2) The corresponding part of the working case in the concept case : { BOOK, PROCEEDINGS, JOURNAL, PUBLICATION, AUTHOR, PAPER, PUBLISHING_COMPANY, IS-A(BOOK,PUBLICATION),

---

[53] Basically, the deleted concept and the relation with the related concept will be packed as an index set. The index set will be partitioned into two or more subsets if there are either two more objects or two more relations in the index pack. The maximum deleting element is the index set itself. In this example, because there is only one deleting concept, there is only one deleting element, which consists of the deleted concept and its relation with the related concept such as BOOK in this example, and is also the maximum deleting element. A more complicated situation, for example with more than one deleted concept, will be dicussed in chapter 5.

IS-A(PROCEEDINGS,PUBLICATION), IS-A(JOURNAL,PUBLICATION), r1(PAPER,AUTHOR), r1(BOOKS,AUTHOR), r1(PAPER,JOURNAL), r1(PAPER,PROCEEDINGS), r1(BOOK,PUBLISHING_COMPANY)}.

(3) The elements in the working case and the corresponding elements in the concept base are in one-to-one correspondence.

(4) The PUBLISHING_COMPANY concept is the deleting concept (the related concept is the BOOK concept).

(5) According to the discussion in section 4.3.2, the index set {PUBLISHING_COMPANY,r1(PUBLISHING_COMPANY,BOOK)} will be transferred into the case base.

(6) In this example, the deleting element is same as the index set: {PUBLISHING_COMPANY,r1(PUBLISHING_COMPANY,BOOK)}.

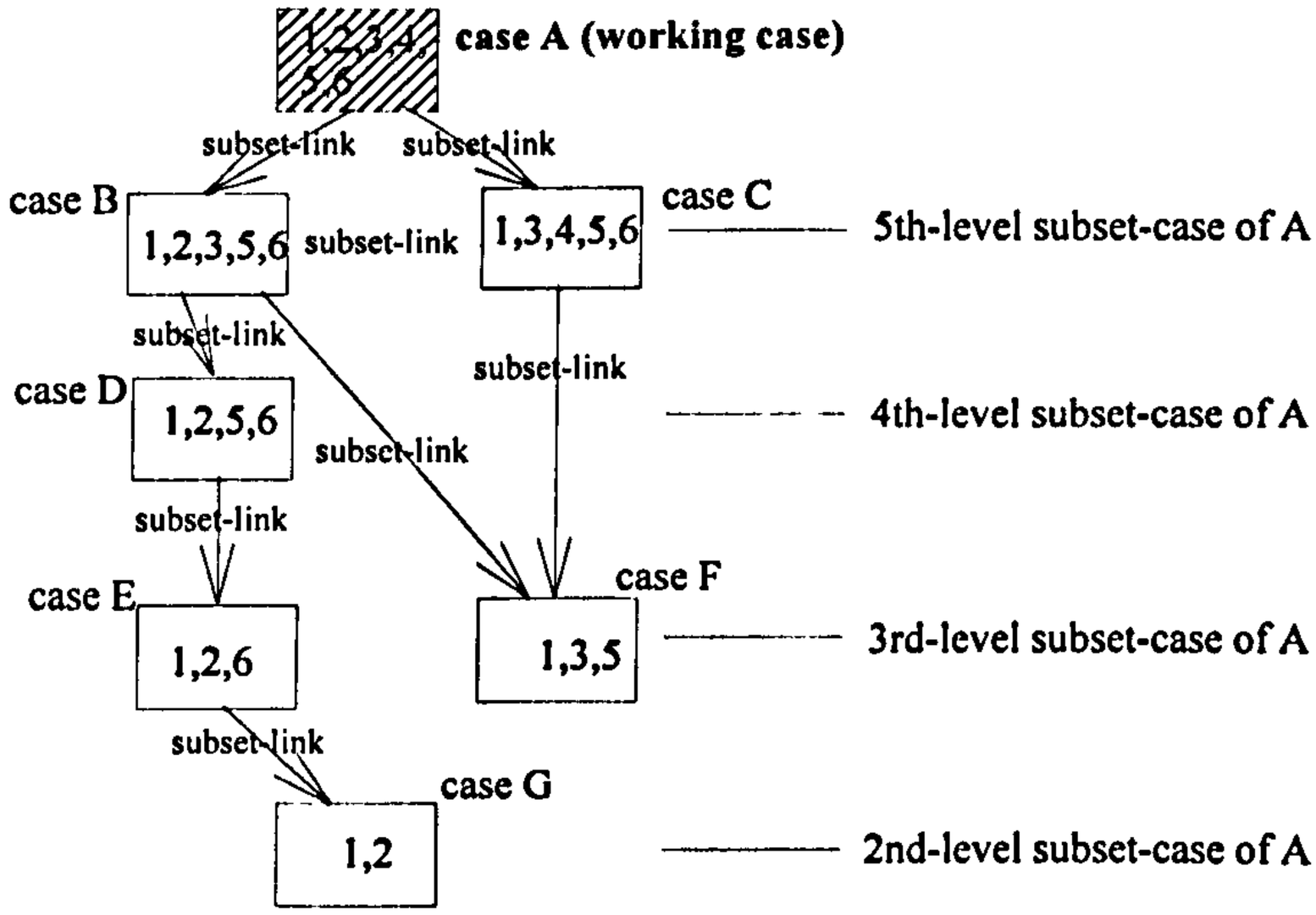
Figure 4.37 : An example of the dual-base knowledge structure

(II) the property for terminating the retrieval process

The subset links of the working case are used to find out the subset-case, which includes the non-deleted entities (relationships) but does not include the maximum deleting element, of the working case. The question encountered here is whether the case base contains a subset-case of the working case which satisfies the user's requirement. The situation can be tackled by the derived property of the subset relation, as illustrated as follows.

Definition: The Kth-level sub-case (s) of one case that is (are) the proper subset (s) of the case and $K \in [m-1,2]$, $m \geq 3$ (where m is the number of the concepts in the case). For example, as shown in Figure 4.38, if case A involves six concepts, then its sub-cases might be possible in four levels. The sub-cases, case B and case C, including five concepts, are called the 5th-level sub-cases of case A; the sub-case, case D, including four concepts, is called the 4th-level sub-case (s) of case A; the sub-cases, case E and case F, including three concepts, are called the 3rd-level sub-cases of case

A; the sub-case, case G, including two concepts, is called the 2nd-level sub-case of case A.



The layers of Kth-level subset-case is just for case A

Figure 4.38 : The 4 layers of subset-cases of case A

Basically, the process of retrieving the relevant subset-case(s) of the working case involves travelling down from the working case until the end. Based on the derived property of the subset relation discussed above, the subset-case of the working case which satisfies the deleting-requirement must be in the (m-k)th level of the k layer tree of the working case. In other words, if there is no subset-case of the working case which satisfies the user's requirement, then the retrieval process can be exited.

4.5.3 The Property of the Exemplar Link

As discussed in section 2.1.2, based on their intended use, the properties of an entity and a relationship between entities might be different. The exemplar link can be used to co-ordinate the processes of retrieving the relevant parts of conceptual schemata in different cases for adapting the role and the content of the concerned entity, as exemplified below.

(I) The aspect of entity role (relationship) ·

The retrieval of the variations of the relationship of entities can be accessed by means of the exemplar links of the relevant entities. These exemplar links are used together to constrain the process of retrieving the varied relationship(s) of the concerned entities in different cases, i.e. the scope of the retrieval is only focused on the varied element of the relevant case, not on the whole case. For example, as shown in Figure 4.39, the process of retrieving the varied relationships of entities A and B in case 3, is illustrated as follows.

Step 1: Clicking the relationship (R1) of the entities A and B in case 3.

Step 2: Based on the one-to-one correspondence, the corresponding concepts, a and b, are highlighted by means of the has-exemplar(a,$A_{case\ 3}$) and has-exemplar(b,$B_{case\ 3}$) links.

Step 3: By means of the three types of links of case 3 - the supset, overlapped (concept-relation level) and subset links - the varied relationships of entities A and B in different cases will be displayed in turn. The displaying

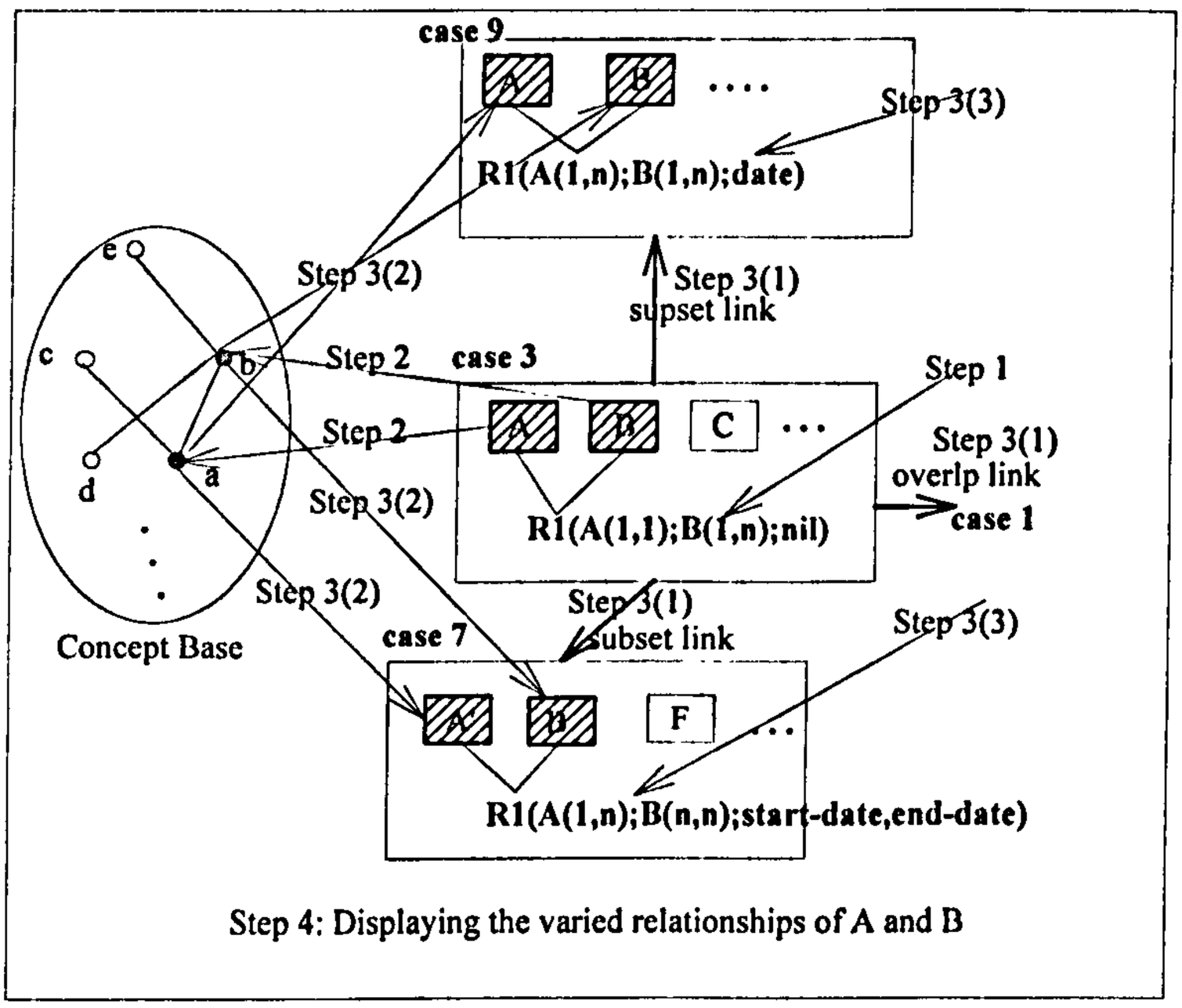
variations are handled by the highlighted concepts and their exemplar links. For example, when the overlapped case 1 has got the relationship of entities A and B (through the overlapped link of case 3),

(1) first, case 1 will be activated and its elements will wait to be highlighted;

(2) the entities A and B will be highlighted by means of the has-exemplar(a, $A_{case\ 1}$) and has-exemplar(b, $B_{case\ 1}$) links;

(3) the relationship of A and B will be activated for comparison with the clicked relationship in case 3.

Step 4: The varied relationship of A and B will be displayed as clues to let users examine whether they need to be considered in the current situation (Repeat step 3).



Step 4: Displaying the varied relationships of A and B

Figure 4.39 : The process of the adaptation of the relationship between entities

(II) The aspect of entity content (attributes)

The retrieval of the variations of an entity can be accessed by the exemplar links of the entity. These exemplar links are used directly to reach the variations of the concerned entity. For example, as shown in Figure 4.40, the process of retrieving the varied contents of the entity A in case 3, is illustrated as follows:

Step 1: Clicking the content (attributes) of the entity A in case 3.

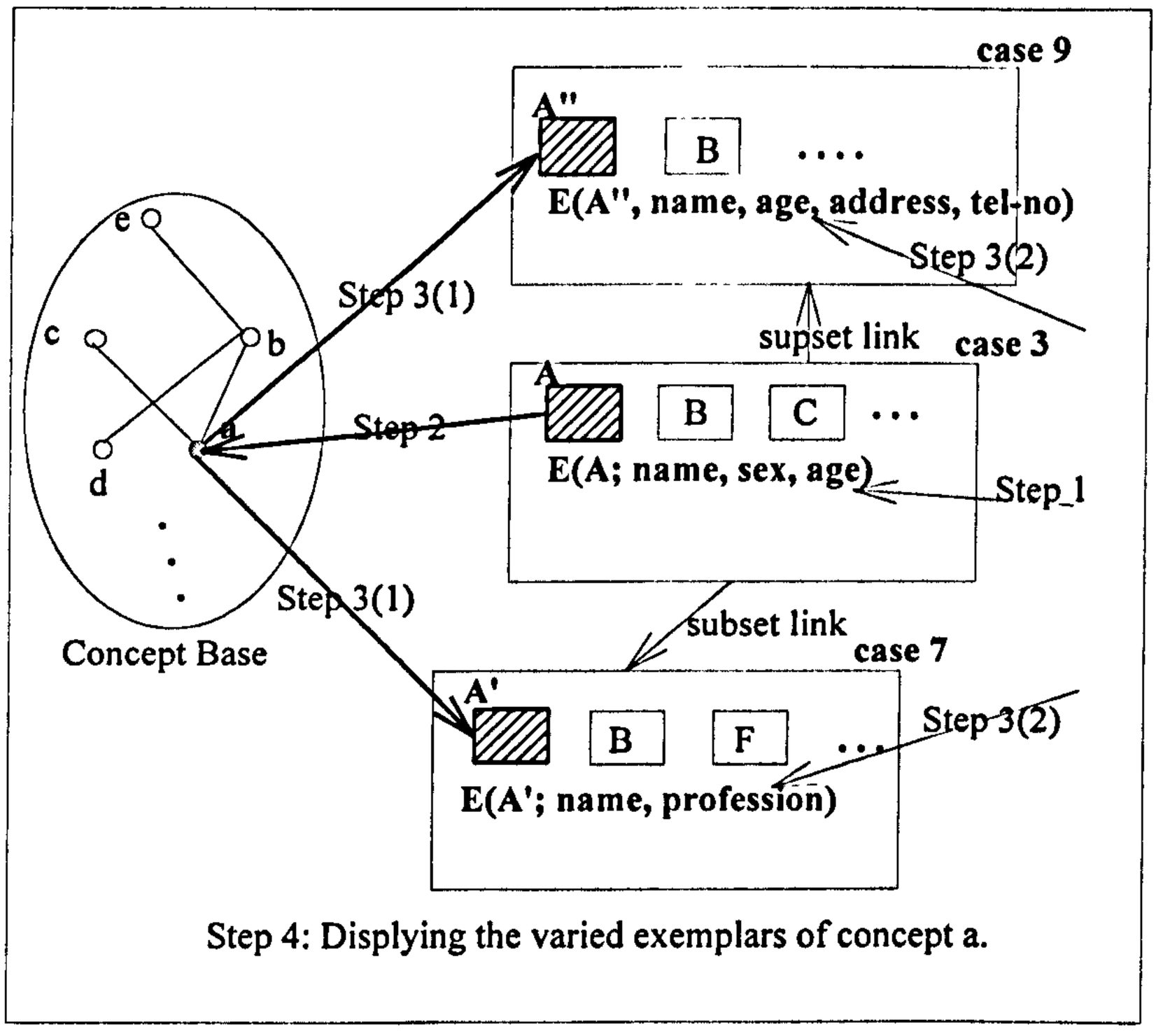
Step 2: Based on the one-to-one correspondence, the corresponding concept, a, is highlighted by means of the has-exemplar ($a, A_{case\ 3}$) link.

Step 3: By means of the exemplar links of concept a, the content of entity A in different cases (or stand-alone entitiy) is activated for comparison with the clicked attributes in case 3. The displaying variations are handled by the highlighted concept and its exemplar links as follows.

(1) The entity A in a different case, such as case 9, will be highlighted through the has-exemplar($a, A_{case\ 9}$) link.

(2) The attributes of the entity A in case 9 are activated for comparison with the clicked attributes in case 3.

Step 4: The varied exemplars of concept a will be displayed as clues to let users examine whether they need to be considered in the current situation (Repeat step 3).



Step 4: Displying the varied exemplars of concept a.

Figure 4.40 : The process of the adaptation of the attributes of an entity

4.5.4 The Property of the Typical Link

Using the is-typical links, the typical system cases are activated by default. Depending upon the user's intentions (the analogous reference or the similar reasoning), there are two ways of using the is-typical links. For the purpose of the similar reasoning, which is a key issue in the DBKM environment, the typical system case is activated as a working case to provide a holistic comprehension of the specified UoD and to serve as a reference point for the retrieval similar cases. This process of retrieving similar cases is more complicated than the process of analogical reasoning, and will be illustrated in chapter 5. This section focuses only on the process of analogous reference.

In order to facilitate analogical thinking, the domain case together with its typical system cases are activated by means of the is-typical links. The help of analogical thinking can be achieved in two ways: the single-unit reference and the whole-structure reference, as illustrated below.

(I) The process of single-unit reference:

The single-unit reference can be discussed in terms of the entity and the relationship aspects.

(1) For the entity aspect (using the transportation centre entity as an example, as shown in Figure 4.41):

(1.1) Based on the one-to-one correspondence, the transportation centre concept in the case base is highlighted by means of the has-exemplar(transportation centre, transportation centre).

(1.2) The station and airport concepts are highlighted by means of the IS-A(station, transportation centre) and IS-A(airport, transportation centre) relations.

(1.3) The exemplars of the station and airport concepts - the content of the station entity in the activated system case Atc1 and the content of the airport entity in the activated typical system case Atc2 - are displayed by means of the has-exemplar(station, station) and has-exemplar(airport, airport).

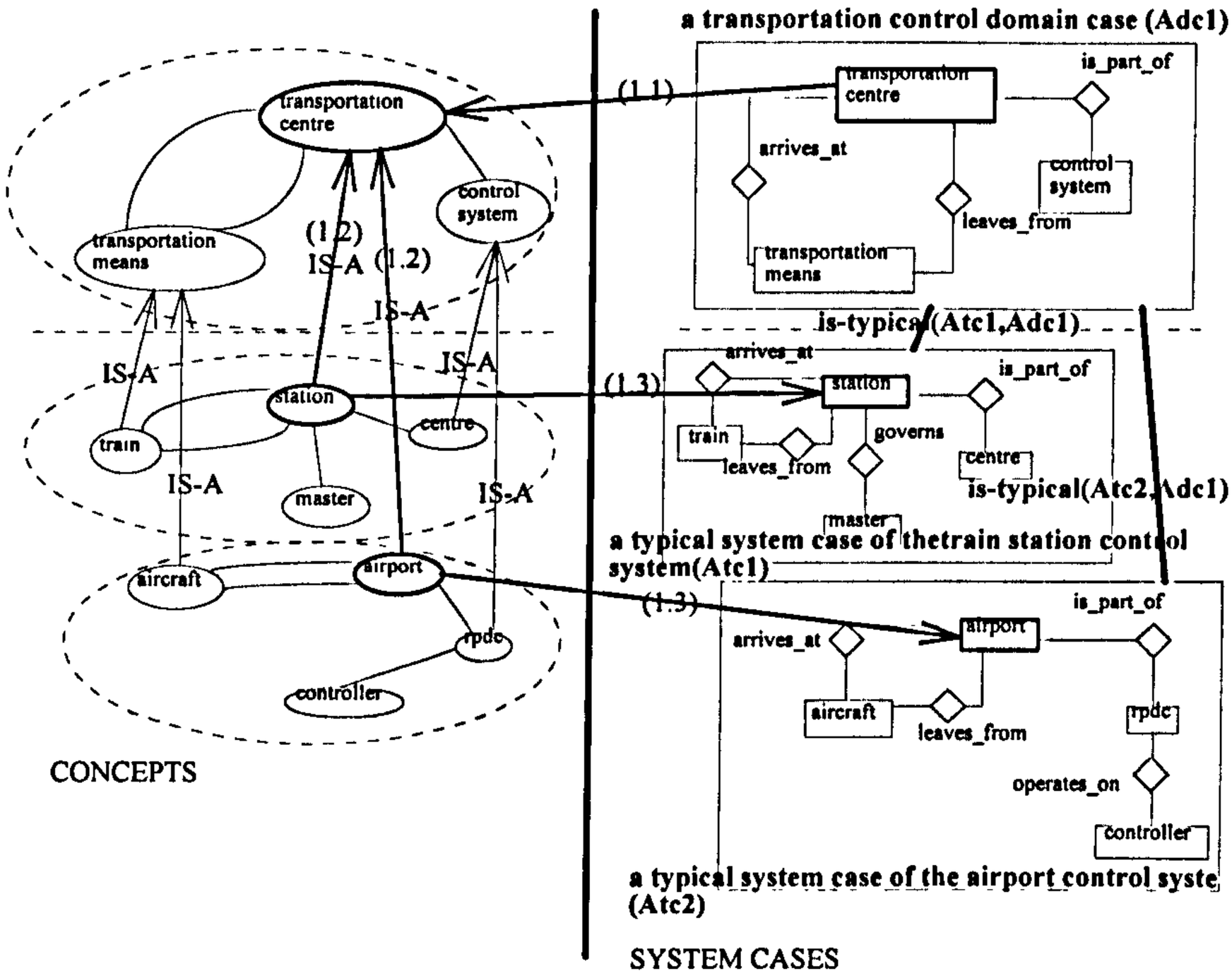


Figure 4.41 : An example of the single-unit reference (the entity aspect)

(2) For the relationship aspect (using the relationships between the transportation centre entity and the transportation means entity as an example, as shown in Figure 4.42):

(2.1) Based on the one-to-one correspondence, the transportation centre concept and the transportation means concept in the case base are highlighted by means of the has-exemplar(transportation centre, transportation centre) and has-exemplar(transportation means, transportation means).

(2.2) (2.2-1) The train and station concepts are highlighted by means of the IS-A(station, transportation centre) and IS-A(train, transportation means) relations; and (2.2-2) the aircraft and airport concepts are highlighted by means of the IS-A(aircraft, transportation means) and IS-A(airport, transportation centre) relations.

(2.3) (2.3-1) The relationships of train and station in the activated system case Atc1 are displayed by means of the has-exemplar(train, train) and has-exemplar(station, station); and (2.3-2) the relationships of aircraft and airport in the activated typical system case Atc2 are displayed by means of the has-exemplar(aircraft, aircraft) and has-exemplar(airport, airport).

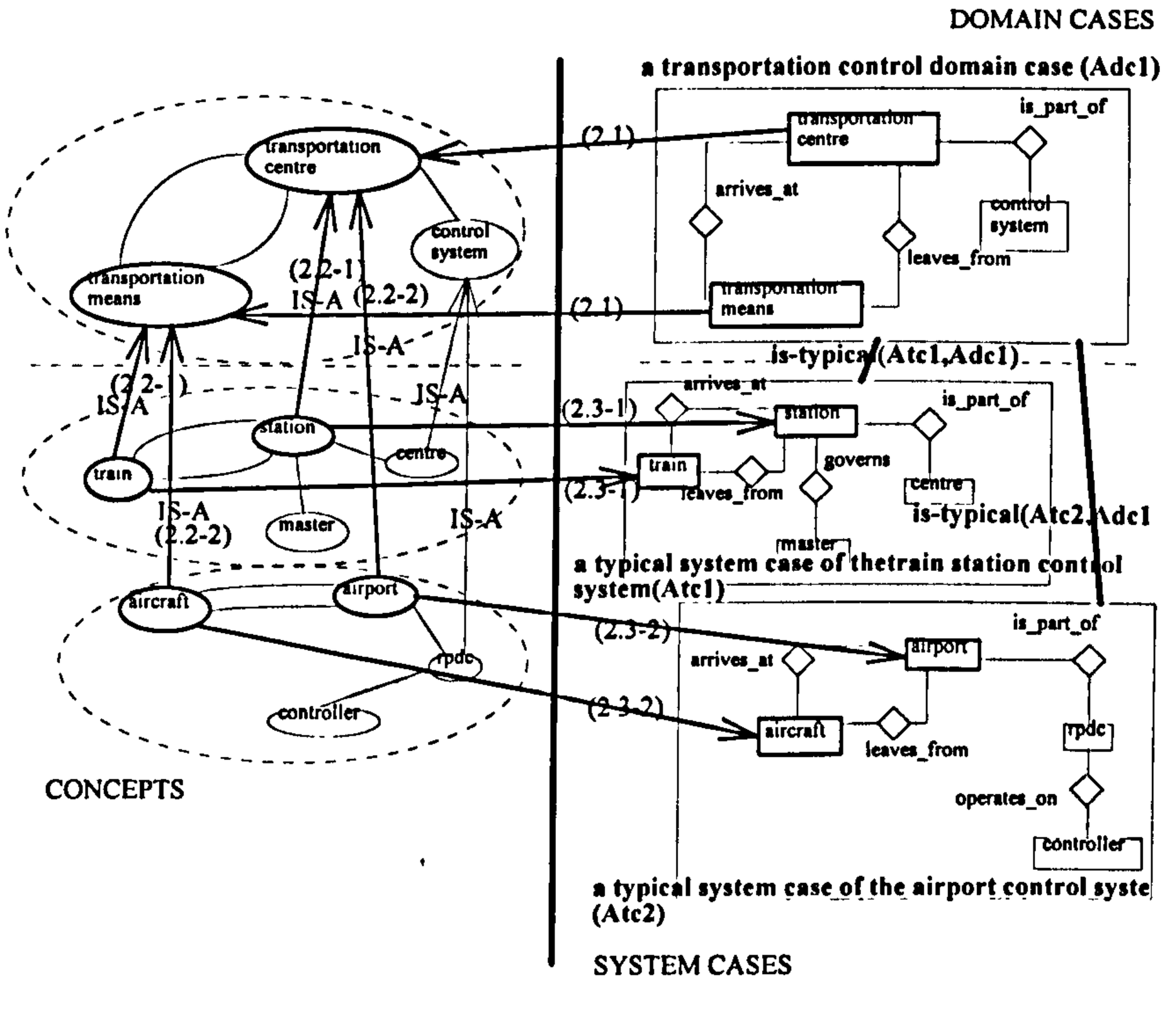


Figure 4.42 : An example of the single-unit reference (the relationship aspect)

(II) The process of whole-structure reference:

The typical system cases of a domain case are displayed directly as whole patterns. The users can refer to these patterns to discover the information which might be used to inspire their thinking. For example, as shown in Figure 4.43, not only the

structures of the typical system cases such as Atc1 and Atc2 cases, but also the contents of entities and relationships in these two cases can be used as materials of anaological reasoning.

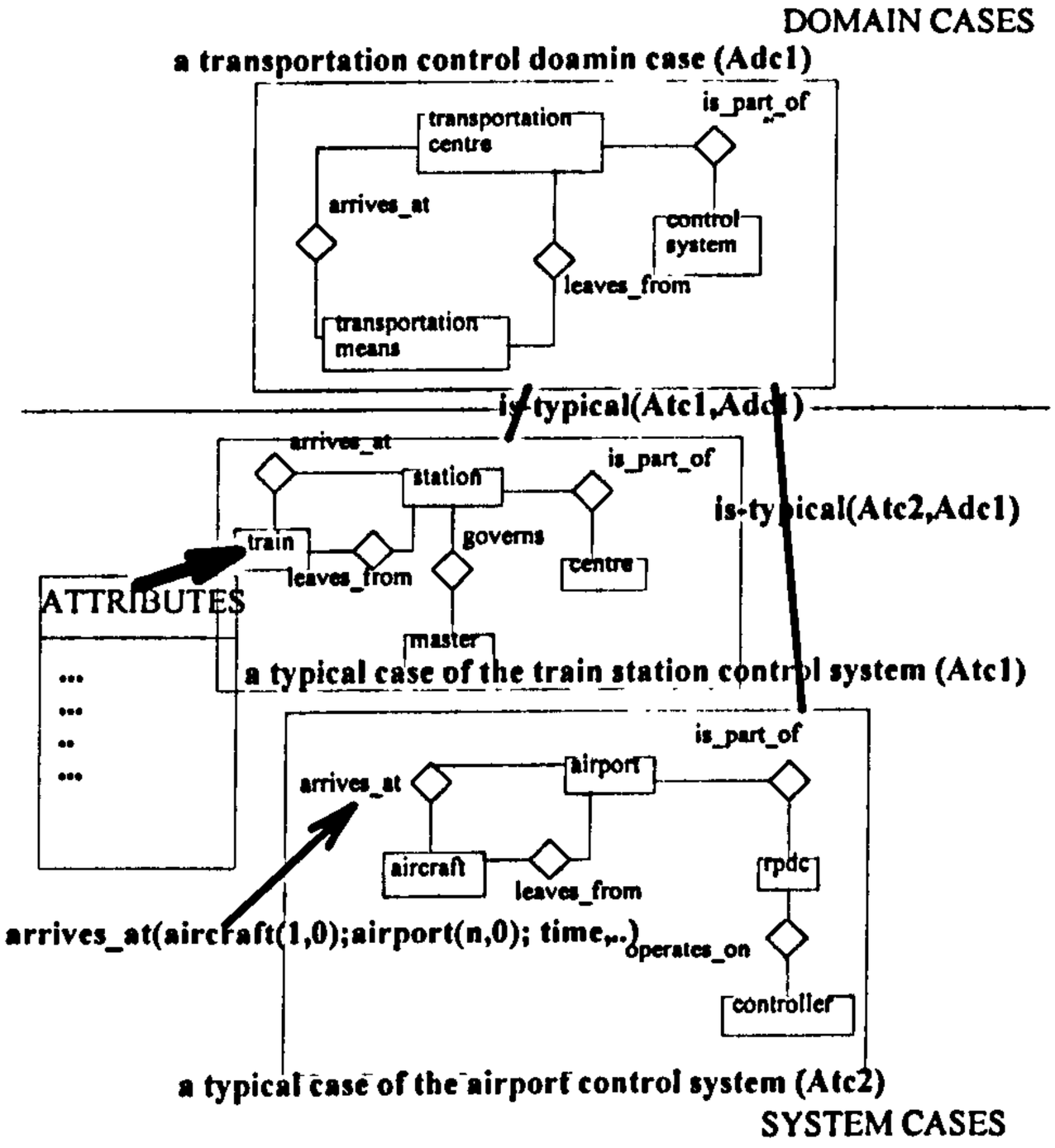


Figure 4.43 : An example of the whole structure reference

Chapter 5

The Process Model of the DBKM

Two main themes are analysed for the process model of the DBKM in order that the finished conceptual schemata or part of them can be reused and so help the difficulties of the conceptualising task. These are *case retrieval* - how to recall the relevant concrete experiences from the case base; and *knowledge accumulation* - how to store the newly gained knowledge in the knowledge model.

The whole process cycle of the DBKM, which is an elaboration of the general process cycle of the CBR paradigm, will first be discussed in section 5.1. The mechanism of case retrieval in the DBKM environment consists of a key recalling process that is carried out by the elaborated sub-process(es). The first part of the key recalling process is supported by the first-pass focus sub-process. The second part is carried out by the recalling cycle, including four sub-processes. There are the second-pass focus, transfer, find and select sub-processes. The third part is for the adapt sub-process based on the notion of case-based substitution. These three parts of the key recalling process will be illustrated in section 5.2. Finally, the mechanism of knowledge accumulation in the DBKM environment is the store sub-process. Through this mechanism, the effects on the DBKM may be considered under two headings: the growth of the case base and the enlargement of the concept base, and will be depicted in section 5.3.

5.1 The Process Cycle of the DBKM

Mednick [1962] found that people with low, flat associations of concepts are more creative than those with high, steep associations of concepts, i.e. the concept associations of the highly creative individual are characterised by fewer stereotypes and less commonality than those of less creative individual.

The DBKM, on the one hand, has a number of cases (concrete and stereotyped[54] experiences) stored in the case base and, on the other hand, has rich vocabularies (general domain object-concepts associated by two simple relations) in the concept base. The latter is a relatively flat and weakly bound structure consisting of the general domain objects and their relations. It is easier to access links between relevant object concepts, leading to remote associates. The former consists of steep associations and strong bounds between related cases respectively comprising the relevant entities and relationships of the specific reality (UoD). It is difficult to retrieve links between relevant cases, thus resulting in the confusion of multiple entry. The conceptual knowledge fundament of the DBKM integrates these two kind of knowledge structure to assist users flexibly to retrieve the appropriate cases by means of the relevance of the two structures.

---

[54] Stereotyped experiences means the fixed general objects (relations) - entities and relationships - which are believed by people of the specific environment to represent particular types of things in a specified UoD.

The CBR paradigm is the pivot of the whole process cycle of the DBKM. Although the general process cycle of the CBR paradigm consists of two processes (the recalling and modifying processes mentioned in section 3.3.2), the method of fleshing out the details of the process cycle is based on wholly pragmatic grounds. There are no universal CBR process models suitable for every domain of application. However, the general process cycle of the CBR can still be used as a guide to develop the process model of the various CBR systems in particular domains and environments, but in their own right.

In the DBKM environment, the designers are initiators of a discourse and have ultimate responsibility for the result, and the DBKM is used to complement the designers' memories. Figure 5.1 illustrates the whole process cycle of the DBKM. As usual with the typical CBR cycle, it consists of two main processes, but since it has a more elaborated recalling process, it includes five sub-processes: focus (first-pass and second-pass), transfer, find and select. The whole cycle is supported by three distinct types of knowledge - general domain object knowledge, the specific domain object knowledge and meta knowledge, and one external knowledge source - the designers. These four components may be discussed briefly as follows:

(1) General domain object knowledge: Knowledge about general object concepts of the domain. 'General Object Concepts' refers to objects of the real world by their names. They might be physical things such as a book, person, teacher, car, etc., or abstract things such as reward/punishment, education, time, income, etc.

They are associated with two types of simple relations, *is-a* and *undefined* relations, as a whole web-like structure called the concept base that is not only an inference resource but also an index framework of the recalling process.

(2) Specific domain object knowledge: A specific domain object of a case is an exemplar of the corresponding general object concept in the concept base. Specific objects characterise 'General Object Concepts' of the specific UoD by their properties and contents. For example, the 'Author' object in a library domain is described by its attributes such as name, profession, etc. and its relationship such as [write] with 'Book' object. In each environment the relevant specified objects are bounded by a variety of relationships based on its culture and concerns as a whole pattern to model the intrinsic structure of data of the environment. All whole patterns that respectively represent their referent - the specific UoD (environment) - are stored in the case base that comprises a number of concrete experiences.

(3) Meta knowledge: Knowledge about how to control knowledge. There are three main types of meta knowledge. The first type is for the recalling process and consists of the eliciting, indexing, transferring, finding and selecting rules that are used to recall the previous relevant cases by means of interaction with the concept base. The second type is for the adapt sub-process in the modifying process and is based on the notion of case-based substitution that is used to replace the unsuitable content or property of the desired entity. The last type is for the store sub-process in the modifying process and consists of the storing

rules that are used to store the newly gained knowledge into the case base or/and the concept base.

(4) External knowledge source: designers are adapters of the retrieval cases as well as initiators of the UoD. They have got a responsibility for the final result. To initiate the specific UoD, designers use the user interface that might be menu-driven or graphic-mode to focus on the intended problem domain. To perform the adapting work, if there are no relevant cases that can be referred to modify the working case, designers can use a user interface to input their knowing of the specified UoD. In this research, the external interfacing work is beyond our immediate concerns.

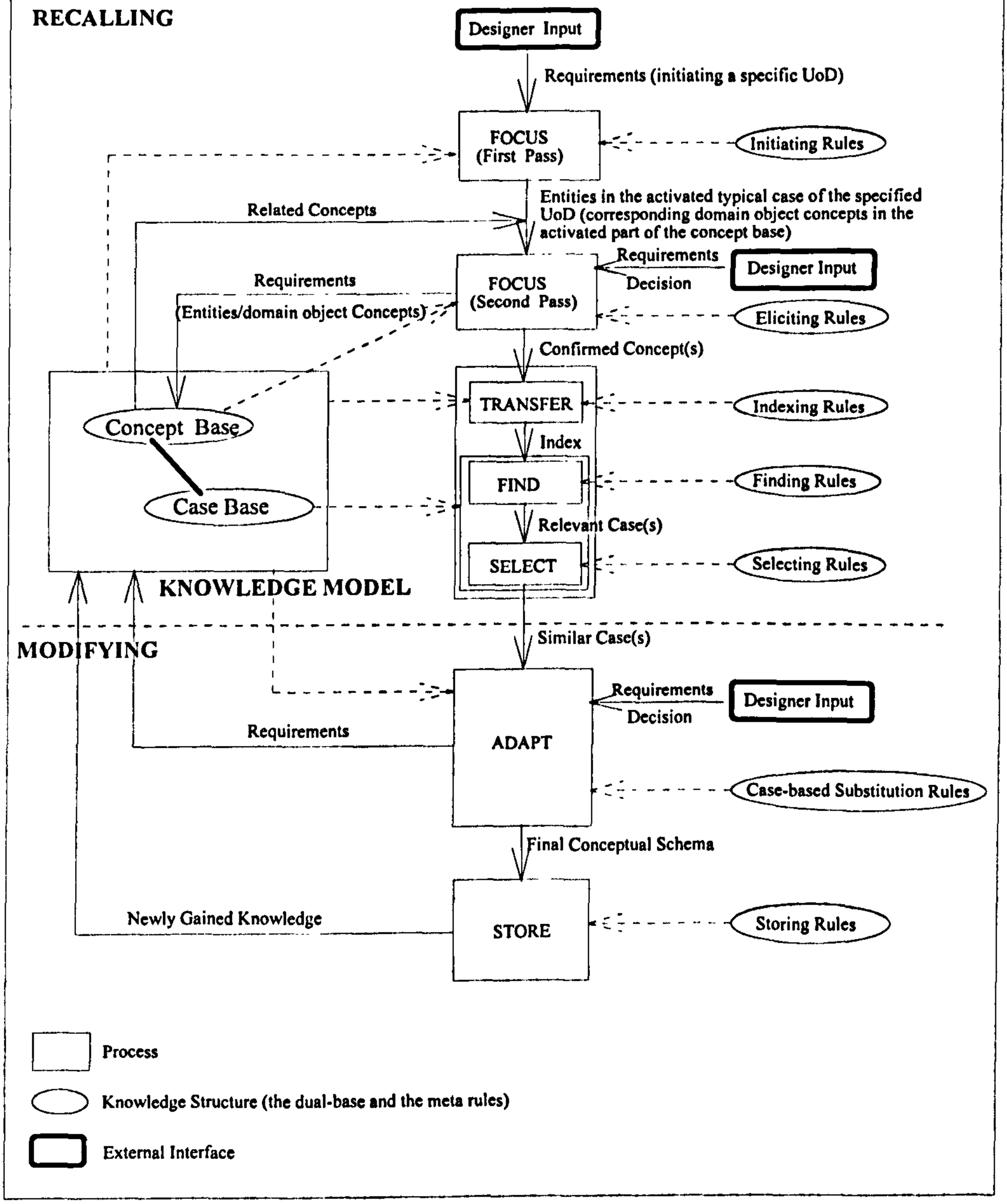


Figure 5.1 : The whole process cycle of the DBKM

5.2 The Key Recalling Process of the DBKM

Williams and Hollan [1981] suggested that the nature of remembering is continually evolving because humans are commonly constrained to use *a partial description*[55] while they want to retrieve the encoded information stored in a very large long-term memory capacity. That is, the relatively general description seeks recover *the appropriate context* that can be used to guide and focus the search for more specific encoded information. For example, knowing about high school in general can help you to elicit the idea of team sports. From that you can get more specific encoded information about football teams, and from this incoming description you can get the specific encoded information about your particular school's football team and then the individuals and the practice venue of the team, and so on. As well as being consistent with the ideas, including the 'conceptual chunking ' [Egan and Schwartz, 1979], and the 'implicit semantics of the mental categories [Chase and Simon, 1972; 1973], discussed before in section 4.1.2, Williams and Hollan further addressed the task of using the partial description to retrieve the relevant encoded information from the long-term memory. These ideas are shared and elaborated by the key recalling process of the proposed DBKM.

The key recalling process of the DBKM is conducted by two agents. First, these are the active designers who are responsible for initiating a specific UoD,

---

[55] The partial description is formed by partial information from the environment and is essentially an indexing mechanism [Williams and Hollan, 1981].

confirming the finding object concepts and adapting the working case. Secondly, these is the passive DBMS agent using the partial descriptions provided by the active designers. The general scenario of this key recalling process is performed by the following episodes: (1) the initiated UoD to activate the appropriate contexts - the typical case of the specified UoD and its corresponding part in the concept base; and (2) the confirmed domain object concepts to form the appropriate context(s) - an index/indices (including the asserted concepts and the relations between/among the displayed concepts) for finding and selecting the relevant concrete experiences in the case base. The concerns of the key recalling process include three issues: (1) how to activate the appropriate context as a based point of further retrieval; (2) how to recall the relevant cases as the candidates of the selected case that satisfy the user requirements; (3) how to retrieve the relevant case that includes the suitable content and property of the wanted part (i.e. the relationship and the attributes of the desired entity) for adaptation. The first issue is supported by the first-pass focus sub-process, the second issue by the recalling cycle in the recalling process and the last issue is by the adapt sub-process in the modifying process. These will now be discussed in more detail:

5.2.1 The First Part of the Key Recalling Process: the first-pass focus sub-process

This is designed to focus the working domain (a specific UoD) in order to activate the typical case of the specified UoD and the corresponding part of the concept base that serve as reference points to retrieve other relevant cases; and then just the entities (domain object concepts) of the activated typical case (corresponding part of the concept base) of the UoD are displayed as entry points in order to form the relevant contexts to guide and focus the direction of recalling. The process of the sub-process is shown in Figure 5.2. Four meta-rules embedded into this sub-process are presented in Appendix A.1: Initiating Rules.

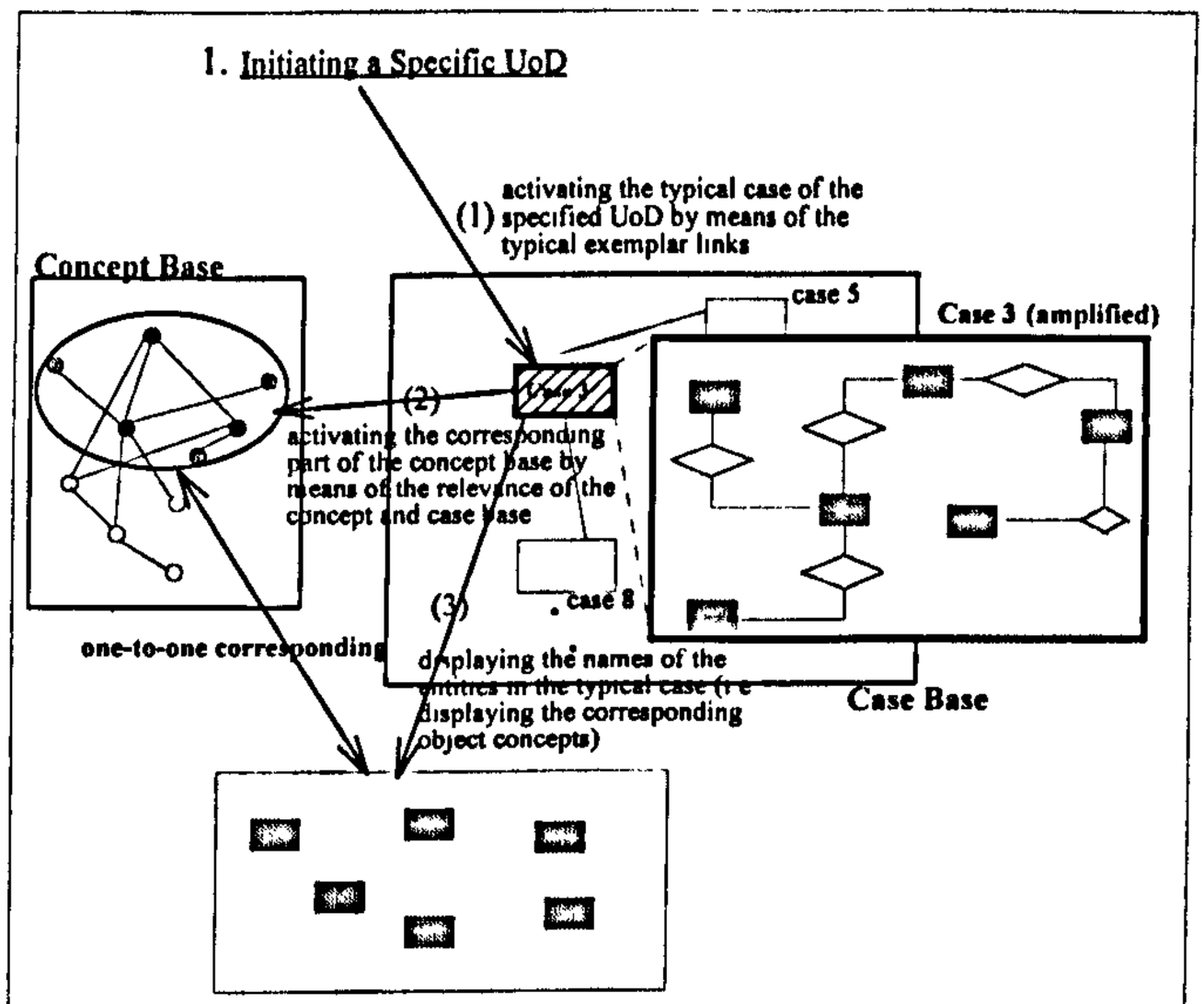


Figure 5.2 : The process of the first part of the key recalling process of the DBKM

5.2.2 The Second Part of the Key Recalling Process: the recalling cycle

Basically, the function of the recalling cycle of the DBKM is to recall the cases that involve the relevant parts of the user requirements. The recalling cycle of the DBKM, as shown in Figure 5.3, is concerned with how the referent cases fitting the user requirements (added or deleted entities) are retrieved.

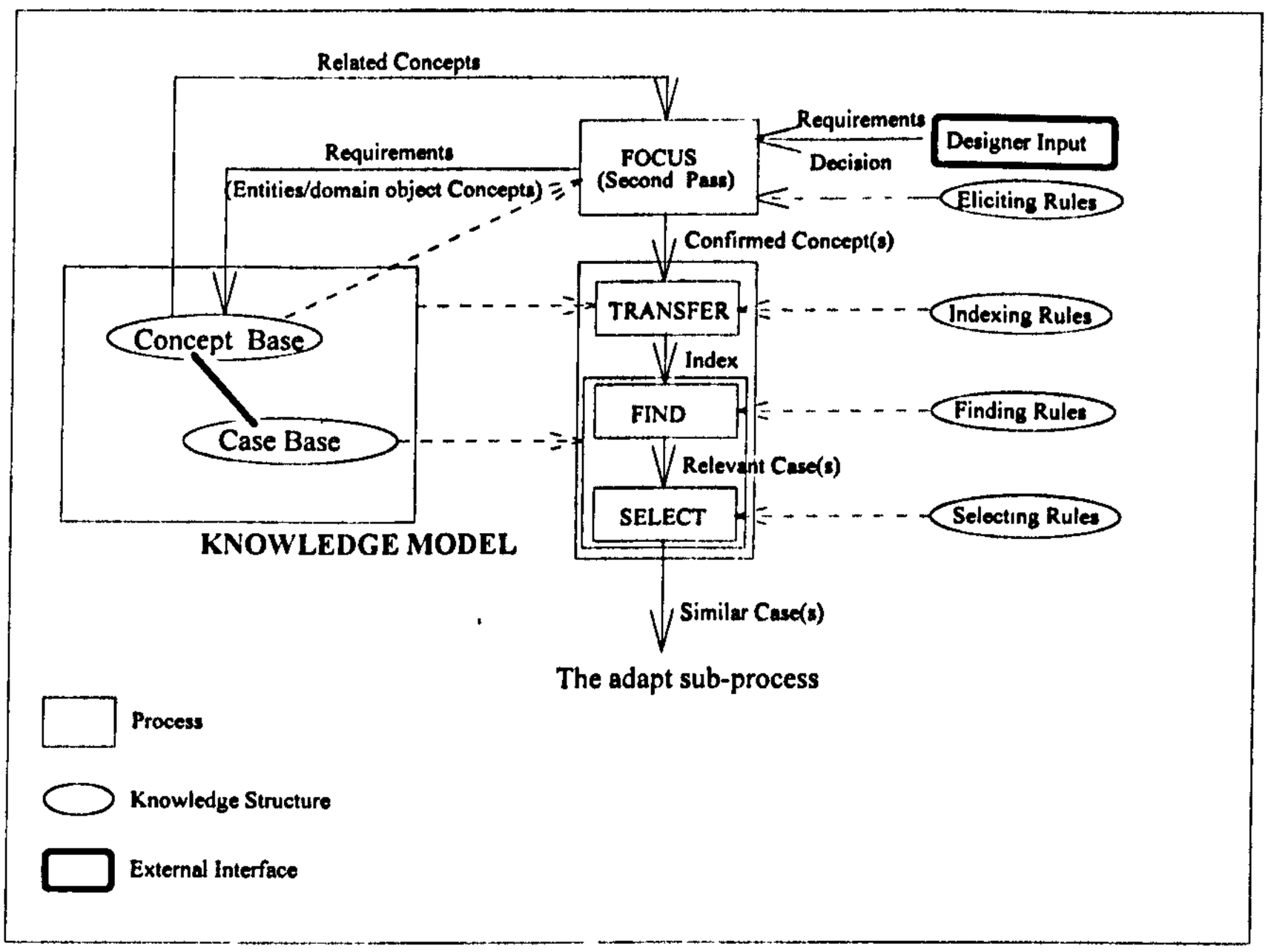


Figure 5.3 : The second part of the key recalling process of the DBKM: The Recalling Cycle

The scenario of the recalling cycle is carried out by the following four sub-processes:

1. **The second-pass focus sub-process** (after the typical case of the initiated UoD and the corresponding part of the concept base have been activated and the entities -corresponding object concepts - are displayed)

This is mainly for the elicitation of the relevant concepts of the user requirements in the concept base. There are two main types of user requirements: the new-added-entity and deleted-entity types. Based on the types of user requirements, two bounded meta-rules, which are presented in Appendix A.2.1: Eliciting Rules, are used in this process. According to the particular requirement type, the relevant concepts will be displayed to let designers confirm them. For example, if there is a new-added-entity requirement, such as adding new entities relating to one (some) of the displayed entities, designers can check whether the desired entities are included in some cases in the case base by means of the displayed concepts (corresponding to the displayed entities) in the concept base in order to investigate whether the desired concepts (corresponding to the desired entities) relate to these displayed concepts. This process, as illustrated in Figure 5.4, is guaranteed by the relevance of the concept and case base discussed in section 4.2.3. The mapping from the displayed entities in the case to the one-to-one corresponding domain object concepts in the concept base is by means of the exemplar links of entities and object concepts.

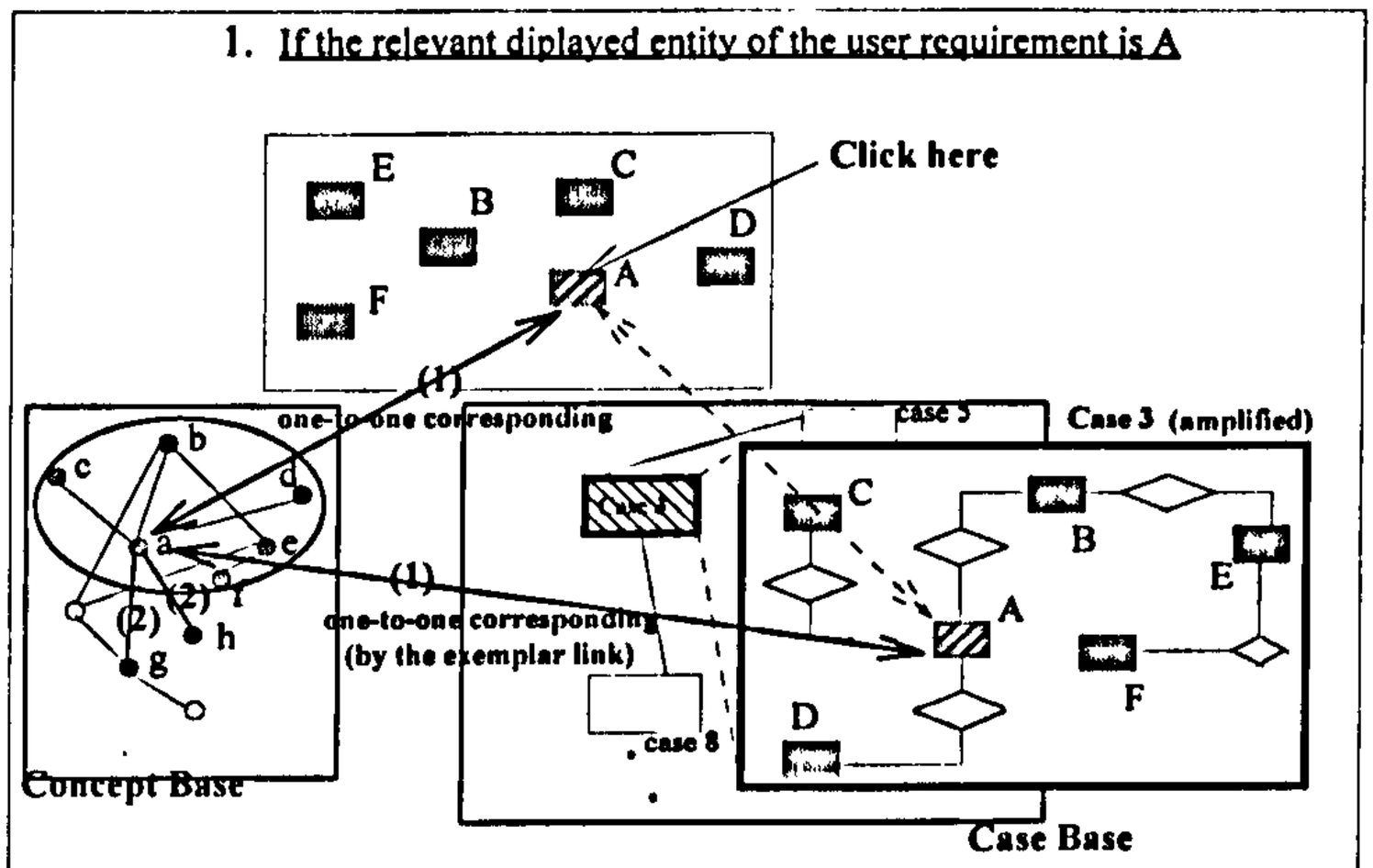


Figure 5.4 : The second-pass focus sub-process of the recalling cycle of the DBKM

2. The transfer sub-process

This aims to pack the confirmed concepts into different types of index based on the requirements. The index is a set of the relevant concepts and the relations with the displayed concepts in the working case. For example, the confirmed concepts, g and h, in the above added-requirement example, as shown in Figure 5.4, will be packed into an index as {g, h, r1(a,g), r1(a,h)}[56] and the index of deleted entity C is {c, r1(a,c);a, b, d, e, f, r1(a,b), r1(a,d), r1(b,e), r1(e,f); Φ}. The forms of index are based on one meta rule(about the new-added-entity type) and another meta rule(about the deleted-entity type) of this process and are presented in Appendix A.2.2: Indexing Rules.

---

[56] If the relation of concept *a* and *g* is an undefined relation, which is undirectional. The r1(a,g) just signifies that there is a relation between concept *a* and *g*.

3. The find sub-process

In this sub-process the concern is only with whether the cases include the relevant concepts (relations)[57] in the incoming index pack. Before finding the relevant case, the index pack might be partitioned into two or more subsets if there are either two more object concepts or two more relations in the index pack. For example, in the new-added-entity type of user requirement, the index pack, {g, h, r1(g,a), r1(g,h)}, will be partitioned into three subsets called the *finding elements*: {g, r1(g,a)}, {h, r1(h,a)}, {g, h, r1(g,a), r1(g,h)} and the finding set is a set of including these finding elements, {{g, r1(g,a)}, {h, r1(h,a)}, {g, h, r1(g,a), r1(h,a)}}, that are used as guides to retrieve the relevant cases. According to the partitioned subsets, the sub-process is guided to find the relevant cases by means of the working case. Obviously, the first priority of the find sub-process is to find the relevant cases that include the relevant concepts as far as possible. In other words, the maximum element (the index pack itself) is a first target of the sub-process, such as the {g, h, r1(g,a), r1(h,a)} in the above instance. The details of how to partition the index pack and how to use these partitioned subsets to find the relevant cases are presented in Appendix A.2.3: Finding Rules which consist of seven meta-rules for the new-added-entity type of user requirements and five meta-rules for the deleted-entity type of user requirements.

---

[57] Although the retrieved overlapped case of the working case includes the maximum element of the finding set, because of the essence of the finding elements that consist only of the concepts and the relevant relations, the contents of the entities and the properties of the relationships of the retrieved case might not be suitable for the user requirements. However, this unsuitable situation can be offset by the case-base substitution strategy discussed in section 5.2.3.

Basically, the finding-sub-process, based on the independent essence of the sub-super and the overlap links, can be carried out through two independent paths. One is, according to the user requirement, by travelling up the supset link of the working case or by travelling down the subset link of the working case to find the cases that satisfy the user requirement. The other is by using the overlapped links of the working case to find the overlapped cases that satisfy the user requirement. After the finding sub-process, those relevant cases are marked and then passed to the next sub-process: the select sub-process. The consequence here is very complicated. For example, in the case of the new-added-entity type of user requirement, the main situations which occur here are as follows: (1) only superset-cases of the working case that include the maximum finding element of the finding set; (2) only some overlapped cases that are of the concept-relation type, of the working case, including the maximum finding element of the finding set; (3) only some overlapped cases that are of the concept type, of the working case, including the maximum element of the finding set; (4) four ($C_3^3 + C_2^3$) ways of combining the above situations; (5) the superset-cases of the working case that together cover the maximum finding element, but none of them include the maximum finding element of the finding set; (6) only the overlapped cases of the working case that together cover the maximum finding elements, but none of them include the maximum finding element of the finding set; (7) some of the finding element(s) in some superset-case and some of the finding element(s) in the overlapped cases of the working case. However, the situations can be tackled by various properties or methods. For example, the situations of the superset cases - situations (1), (4), (5) and (7) - can be resolved by the inherent properties of the

supset relation discussed in section 4.5.2.1. In other words, the number of the relevant superset-cases can be restrained by them. The situations of the overlapped cases - situations (2), (3), (4), (6) and (7) - can easily be remedied by the priority of the two types of the overlapped relations, i.e. the overlapped relation of concept-relation type takes priority over the overlapped relation of concept type.

4. The select sub-process

The select sub-process filters out some marked relevant cases. The effect of choosing the marked cases is influenced by the paths of the finding-sub-process which leads to different consequences. Therefore, the heuristics of activating the marked cases, if the three parallel paths - the path for the superset case, the path for the concept-relation overlapped case and the path for the concept level overlapped case - are adopted in the finding-sub-process, might be classified into below.

(1) Only one marked case satisfies the user' requirement (i.e. the maximum finding element or the maximum deleting element): The marked case will be selected.

(2) The marked cases include superset case(s) and the overlapped case(s) which all include the maximum finding element: The marked superset case(s) will be selected.

(3) The marked superset cases which together cover the maximum finding element: The marked superset cases will be selected.

(4) The marked cases include superset case(s), which does(do) not cover the maximum finding element, and the overlapped case(s), which involve the maximum finding element: The marked superset case(s) will be selected and the marked overlapped case which is of the maximum overlapped degree will be selected.

(5) Only the overlapped cases of the working cases have got the finding element and some of them include the maximum finding element: The maximum overlapped degree of the overlapped marked case involving the maximum finding element will be selected.

(6) Only the overlapped cases of the working cases have got the finding element and they together cover the maximum finding element:

Condition 1: if the concept-relation overlapped cases together can cover the maximum finding element, then the marked overlapped cases will be selected in the order of decreasing of the overlapped degree.

Condition 2: if only the concept level overlapped cases together cover the maximum finding element, then the marked overlapped degree cases will be selected in the order of decreasing of the overlapped degree .

Condition 3: if the concept-relation level overlapped case(s) and the concept level overlapped case(s) together cover the maximum finding element, the selections of the two types of the marked overlapped cases are based on the order of the overlapped degree.

(7) The marked cases include superset case(s) and the overlapped case(s), which together cover the maximum finding element:

1. The superset case(s) involving the finding element(s) will be selected.
2. The overlapped case(s) involving the rest finding element(s) will be selected based on the criteria discussed in (5) (except the content of the maximum finding element is replaced by the rest finding element(s)).

(8) The marked subset cases which together cover the maximum deleting element: The marked subset cases will be selected.

However, because the marked relevant cases might be either the overlapped case(s) of the working case or the supset-cases of the working case for the new-added-entity type of user requirement (the subset-cases of the working case for the deleted-entity type of user requirement), the characteristics of the activated marked cases (i.e. the non-filtered marked cases) must satisfy the following criteria:

(1) The activated marked cases must fully cover the user requirements.

(2) If the activated, marked relevant case is of the overlapped case of the working case, the case must be selected in the order of decreasing of the overlapped degree, based on the finding element.

(3) If the activated, marked relevant case is of the supset-case of the working case, the case must be of the supremum (the least upper bound) based on some finding element.

(4) If the activated, marked relevant case is of the subset-case of the working case, the case must be of the infimum (the greatest lower bound) based on some deleting element.

5.2.3 The Third Part of the Key Recalling Process: the adapt sub-process

Basically the function of the adapt process in the DBKM is that of case-based substitution. The notion behind case-based substitution is that the case can suggest a useful alternative, but it may not provide the best overall match to the new situation [Kolodner, 1993]. In the DBKM environment, while the non-filtered similar cases are displayed, the designer selects the appropriate case(s) called the selected case(s) including the wanted parts. By using the interface tool, the wanted part(s) in the selected case(s) are combined into the working case as the initial conceptual data schema. The cases behind the initial conceptual data schema are the working case and the selected case(s) that include some relevant elements in the initial schema respectively. Thus, if the content and the property of the wanted part (i.e. the relationship and the attributes of the desired entity) are not suitable for the user requirements, the designer can use the initial schema, by means of the behind cases, to find out whether other cases involve the suitable content or property of the desired entity. In other words, in this research, the process of adaptation only concerns whether other cases include the suitable part and is not concerned much with the rest of the referent cases. This sub-process is shown in Figure 5.5. Two meta-rules of the adapt sub-process are presented in

Appendix A.3: Adaptation Rules - one is for adapting the entity property (relationship) and the other is for the entity content (attributes).

After the adaptation, the modified schema will be used as a new working template. The temporal link(s) will be created from the new working template to the behind cases[58] in the event that the relevant cases involving the new user requirement can be retrieved in the later design session.

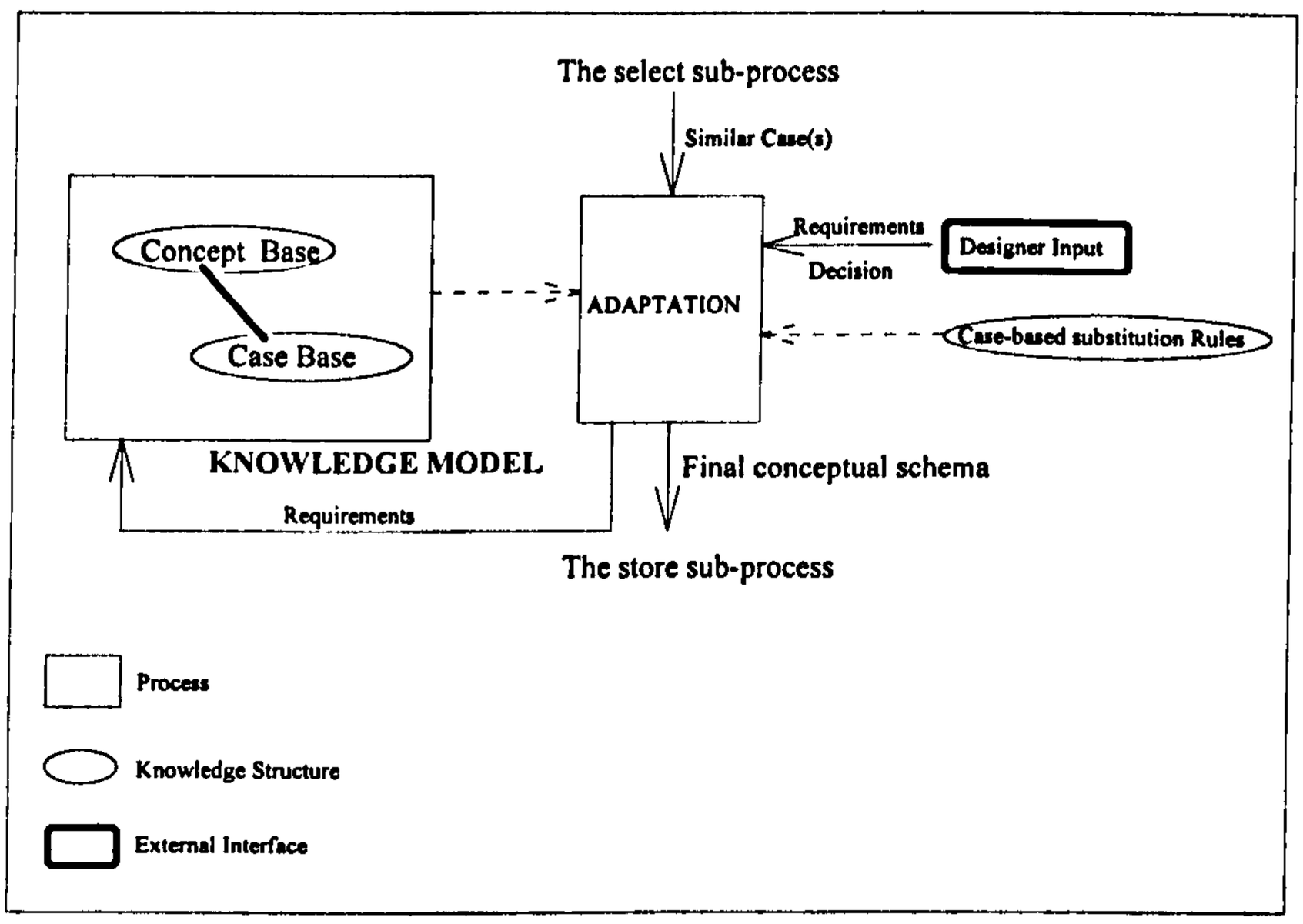


Figure 5.5 : The third part of the key recalling process of the DBKM : The adapt sub-process

[58] The contents of the modified schema might not be just the same as those of the working case and the selected case(s), but from the view of the concept base the main body of the case is just represented in terms of the corresponding concepts and the relations among them. Therefore, the temporal links created from the new working template to the behind cases, but not the relevant cases involving the suitable contents of the desired entity, are also applicable.

5.3 The Effects of Knowledge Accumulation on the DBKM

Learning ability is a desired property of a good knowledge model. The DBKM consists of a concept base and a case base, which can be considered to be complementary to each other. The concept base as a whole represents the general domain knowledge about the real world, and the cases in the case base are the concrete experiences obtained by applying the general domain knowledge to a specific domain.

The evolution of concepts from experiences with exemplars must be a fundamental phenomenon [Medin and Schaffer, 1978]. In this theoretical architecture, through the mechanism of the knowledge accumulation, not only is the structure of the general domain object knowledge expanded by adding new object-concepts accompanying their exemplars or by adding new exemplar[59] of the known object-concept, but the structure of the specific domain object knowledge is enlarged by collecting more larger patterns(cases). While the conceptual schema is finished, it might involve the new knowledge. The effects of the knowledge accumulation on the DBKM are projected into the conceptual knowledge structure of the DBKM: (1) the growth of the case base - the case base increased by gaining more concrete experiences (cases); (2) the enlargement of the concept base - the concept base expanded by enlarging the general domain

---

[59] The new exemplar of a concept means some content of an entity can not be found from the stored cases.

knowledge. The impacts upon the conceptual fundament are considered and influenced by following three aspects : (1)from the view of the concept base, the finished schema involves either a new concept or a new relation between unrelated known concepts, (2) from the view of the case base, the finished schema involves a new property of the relationship between known related entities. There are two situations occurred here. The first one is when the other entities and relationships of the schema are all same with the selected stored case. The second situation is the other entities and relationships of the schema are from the different selected cases respectively. (3) from the view of the concept base, the finished schema just involves a new exemplar of the known concept in the concept base. Therefore, for the DBKM agent:

(1) under the first aspect and the second situation of the second aspect, the finished schema will be stored as a new case for reuse in later session. There are thirty-three meta-rules in the situation. Eighteen meta-rules for the first aspect are presented in Appendix B.1 and fifteen meta-rules for the later are presented in Appendix B.2.

(2) under the first situation of the second aspect, the new property of relationship of the known entities will be annotated to highlight that there is another relationship between these entities in the selected case. The corresponding meta-rule of this situation is presented in Appendix B.3.

(3) under the third aspect, the non-case entity and the exemplar link of its corresponding known concept will be build. The corresponding meta-rule bounded of this situation is presented in Appendix B.4.

Chapter 6

Example and Discussion

Following Chapter 4 (the theory of DBKM) and Chapter 5 (the process model of the DBKM), in this chapter a simulated example is given in section 6.1 to exemplify how the relevant cases are retrieved and the new knowledge is stored, and to provide a comprehensive view for understanding the function of the proposed knowledge model. In section 6.2, the distinctions between the DBKM and other approaches are discussed first from the following viewpoints: (1) for the knowledge-based system, and (2) for the knowledge modelling, emphasising the contrast in the current progress of the conceptual data modelling work. Then the limitations of the DBKM are evaluated in two respects: (1) the static aspect of the UoD, and (2) the passive generalisation of the domain cases and their corresponding concepts and relations.

6.1 The Simulated Example

For clarification of how the relevant cases can be retrieved from the DBKM and how the newly gained knowledge can be retained in the DBKM, the process mechanisms of the DBKM illustrated in Chapter 5 will be explained by an example. Supposing the structure of the concept base and the relations among the

system cases in the case base are originally as shown in Figures 6.1 and 6.2 respectively. To simplify Figure 6.2, only the entities of the case represented as rectangular are listed in the corresponding boxes such as the A,B,C,D,E,F,G,H entities of case 3.

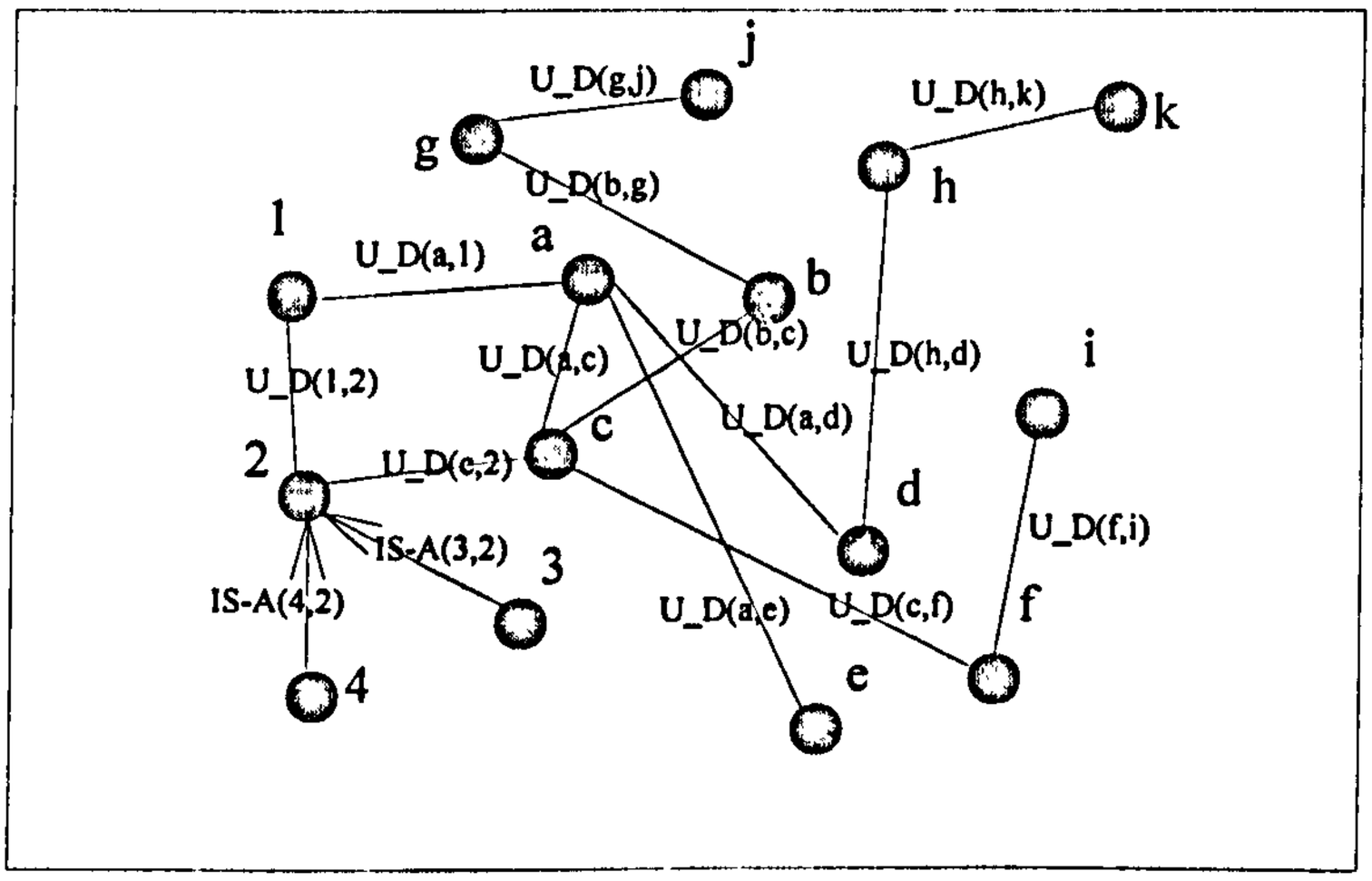


Figure 6.1 : The structure of the concept base (before)

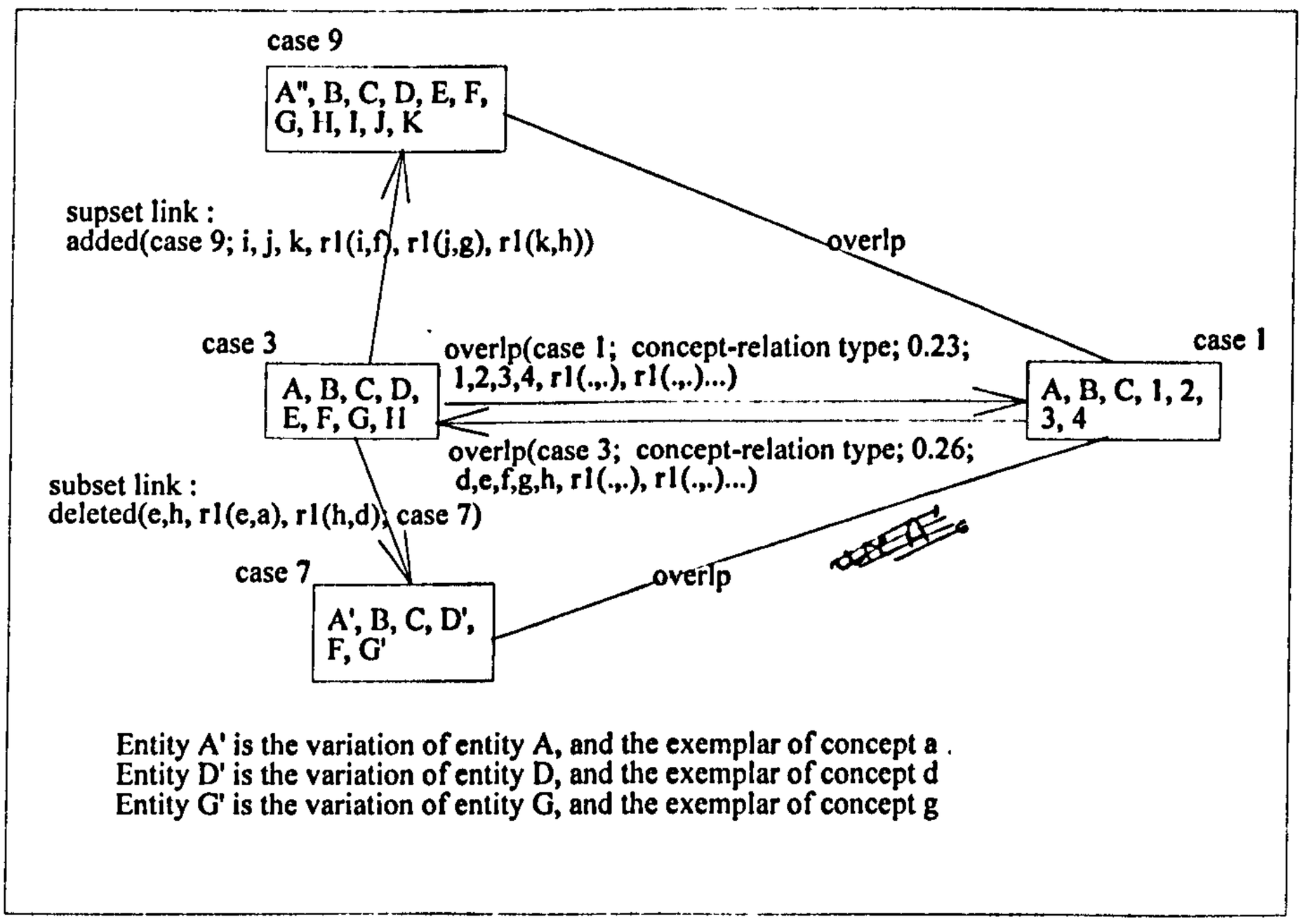


Figure 6.2 : The relations among the system cases in the case base (before)

6.1.1 The Case Retrieval Mechanism: The Key Recalling Process

6.1.1.1 The first-pass focus sub-process

When a specific UoD is initiated, the typical case, for instance case 3, of the specified UoD is activated as the *working case* and then the corresponding part of the concept base is also activated by means of the relevance of the concept and case base. The entities of case 3, for instance A, B, C, D, E, F, G, H entities (in the foreground), and the corresponding objects of case 3 in the concept base, for instance a, b, c, d, e, f, g, h objects (in the background), are used as entry points to retrieve further relevant cases.

6.1.1.2 The recalling cycle

1 The second-pass focus sub-process

(1) Click entities F and G, indicating that the designer wants to add new entities and relationships related to F and G to the new design. This requirement will then be sent to the concept base to find out whether there are concepts connected to the concepts corresponding to entities F and G.

(2) Suppose that concepts i and j are retrieved and are shown to the designer for confirmation.

2 The transfer sub-process

If confirmed, an index set {i, j, r1(i,f), r1(j,g)} will be packed and passed to the next process: the find process.

3 The find sub-process

Before finding the relevant cases, the incoming index pack will be partitioned into two or more subsets, called the *finding elements*, if there are either two or more object concepts or two more relations in the index pack. In this example, the find sub-process is as follows:

(1) By forming the finding set, including three finding elements, {i, r1(i,f)}, {j, r1(j,g)}, {i, j, r1(i,f), r1(j,g)}), as clues to retrieve relevant cases in the case base.

(2) According to the policy of the find sub-process, the finding will be carried out by using the supset and the overlapped links of the working case. Case 9 is marked.

4 The select sub-process

(1) Because of fully covering the user requirements, the marked case 9 is selected and displayed to allow the designer to refer to it.

(2) If there are other user requirements, the cycle will be repeated from process 2.1 to process 2.4. In this example, the new requirement is to add the new entities, for instance entities 1 and 2, and case 1 is another selected case.

6.1.1.3 The adapt sub-process

(1) The relevant parts of the selected cases will be combined manually into the working case as the *initial conceptual data schema.* In this example, the entities in the initial schema are A, B, C, D, E, F, G, H, I, J, 1, 2.'

(2) If the content and property of the desired entity in the initial schema are not suitable for the user requirements, the notion of case-based substitution is used to find out whether other cases involve a suitable content or property of the entity. In the DBKM, there are two aspects to which the case-based substitution is applied.

(I) the aspect of entity role (relationship):

The variations of the relationship(s) of the relevant entities, if any, can be displayed by means of the two links: (1) *overlp* links - using the concept-relation type of the overlp link in the referent case to check whether there is a case including the varied relation of the corresponding concepts . If any, then the overlapped case is activated for further examination; (2) *supset (subset)* links - if no varied relationship can be found by the overlp links, then the supset (subset) links of the selected case are traversed to investigate whether there is a varied relationship in the superset cases (subset cases). The process of case-based substitution - the modification of the relationship - is illustrated in the meta-rule G-Rule 6.1. If no relationship under the variation is found or no satisfied relationship is ready for selection, the designer is allowed to enter a new relationship of the desired entities directly. In this example, case 7 has a

varied content of relationship between entity A and entity C. Now we suppose that the relationship between entity A and entity C fits the user requirements.

(II) The aspect of entity content (attributes):

In this phase, the designer will have the chance to review the contents of each entity in the initial schema, and, if necessary, to modify the contents of entities. Suppose that the designer wants to modify the contents of entity A in the initial schema, then the set of possible attributes, if any, under the variations of entity A in other cases will be displayed for selection. The retrieval of these attributes is done by referring back to the concept base and then traversing the exemplar links to reach the variations of entity A. The process of case-based substitution - the modification of the attributes of the entity - is illustrated in the meta-rule G-Rule 6.2. If no attribute under the variation is found or no satisfied attribute is ready for selection, the designer is allowed to enter new attributes directly. In this example, case 7 has a variation of entity A. Now we suppose that entity A is modified and represented as A'''. Entity G and D are also modified and represented as G'' and D' respectively.

6.1.2 The Knowledge Accumulation Mechanism: The Store Process

In this example, the impacts upon the conceptual fundament of the DBKM are influenced by the following outcomes of this design session:

6.1.2.1 There is a new unknown entity (from the view of the DBKM) in the final conceptual data schema

In this situation, the final schema will be retained as a new case in the case base. Under the general meta-rule G-Rule 7.1, the relevant cases will be located in order to place the new case into the appropriate location in the case base; the corresponding concept node and relation(s) among relevant concepts will be created in the concept base in order that this new concept can be used as the recalling point for retrieving the new case; and the exemplar links of the corresponding concepts and entities will be built in order to supply the paths of whole process of the DBKM . In this example, if the new unknown entity is M and its related entity is entity F, the entities in the final schema are A, B, C, D, E, F, G, H, I, J, 1, 2, M. After the aforementioned design session, the structure of the concept base, the relation structure of the case base, and the relevance of the new stored case (case 14) and its correspondence in the concept base are exemplified and shown in Figure 6.3, Figure 6.4 and Figure 6.5 respectively.

(1) The concept base is enlarged by adding a new concept node and its relevant relation - the m object concept and the U_D(f,m) undefined relation.

(2) The case base grows by retaining a new case - case 14.

(I) Under meta-rule G-Rule 7.1.1.6 (the selected case 9, which is a supset-case of the working case - case 3, is not of the proper subset of the final schema, but includes the maximum finding element - {i, j, r1(i,f), r1(j,g)}) and under meta-rule G-Rule 7.1.1.10 (the selected case 1, which is the overlapped case of the working case - case 3, is not of the proper subset of the final schema, but

includes the maximum finding element - {1, 2, r1(a,1), r1(c,2)}), the relations between the new case and its relevant cases are located as follows:

The selected cases, case 9 and case 1, are all of the overlapped cases of the new stored case - case 14. The working case, case 3, is of the direct subset-case of the new store case, case 14.

(II) Under the modified rational model discussed in section 4.3.3, the overlapped degree of the overlapped cases (case 14 - case 9 and case 14 - case 1) is calculated as follows :

the overlapped degree of the overlapped case from case 14 to case 9 is

$$D_{\text{overlp}}(case\ 14, case\ 9) = \frac{19/25}{1+2/21} \approx 0.69$$

the overlapped degree of the overlapped case from case 9 to case 14 is

$$D_{\text{overlp}}(case\ 9, case\ 14) = \frac{19/21}{1+6/25} \approx 0.73$$

the overlapped degree of the overlapped case from case 14 to case 1 is

$$D_{\text{overlp}}(case\ 14, case\ 1) = \frac{9/25}{1+5/14} \approx 0.27$$

the overlapped degree of the overlapped case from case 1 to case 14 is

$$D_{\text{overlp}}(case\ 1, case\ 14) = \frac{9/14}{1+16/25} \approx 0.39$$

(3) The one-to-one correspondence of the entities of case 14 and its corresponding object concepts in the concept base are as follows: A''' - a, B - b, C - c, D' - d, E - e, F - f, G'' - g, H - h, I - i, J - j, 1' - 1, 2 - 2, M - m.

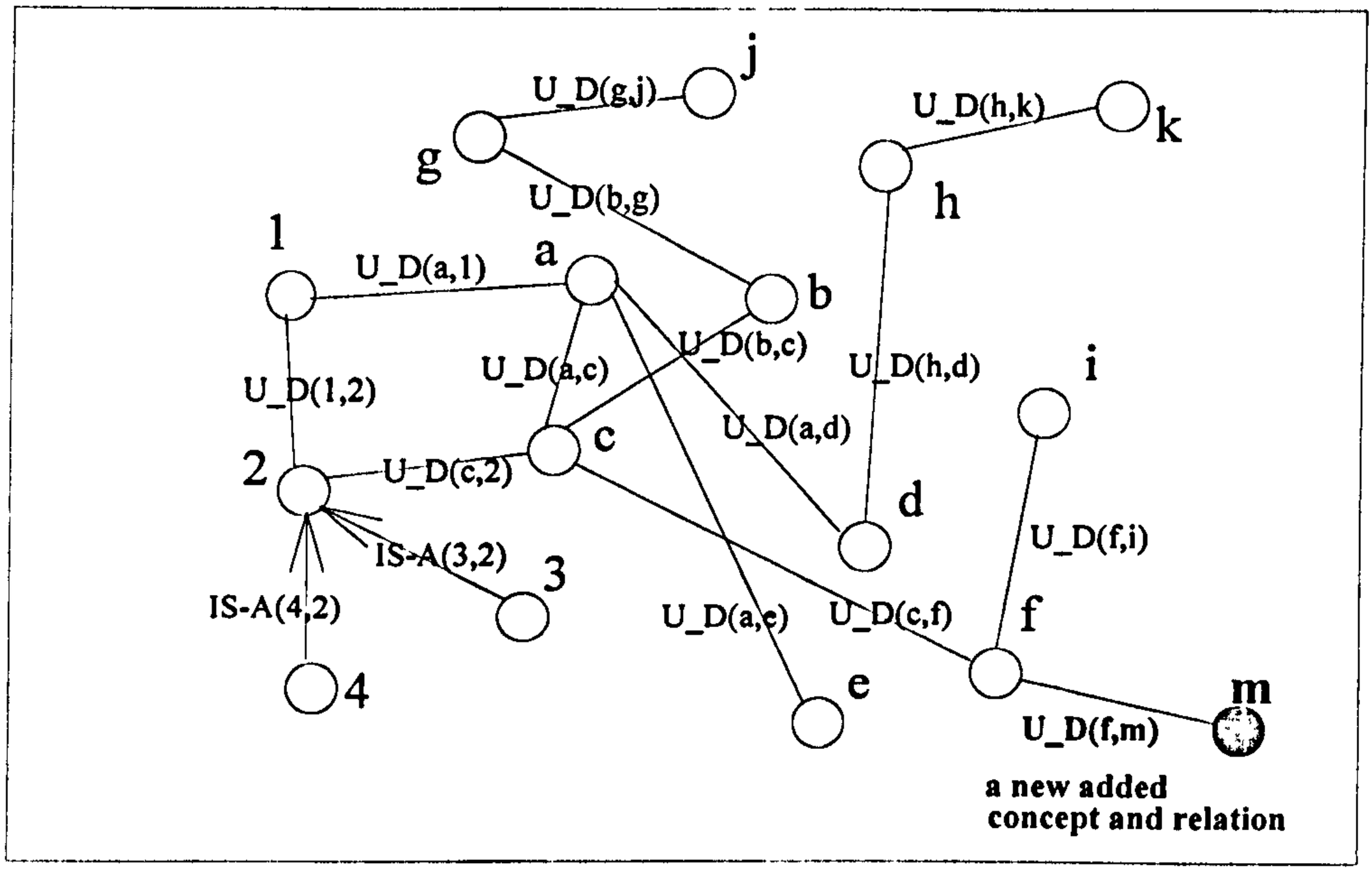


Figure 6.3 : The structure of the concept base (after)

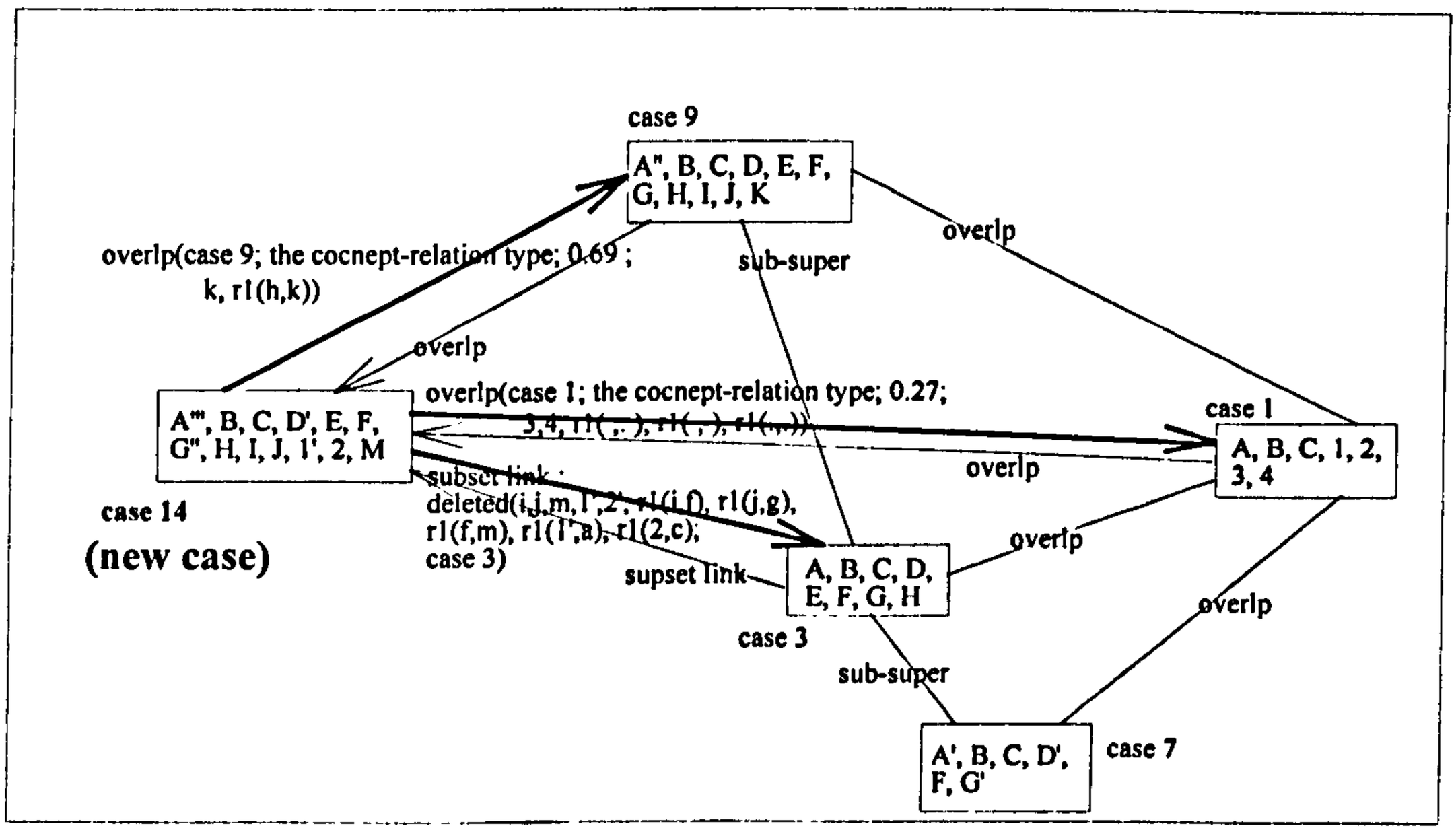


Figure 6.4 : The relations among the system cases in the case base (after)

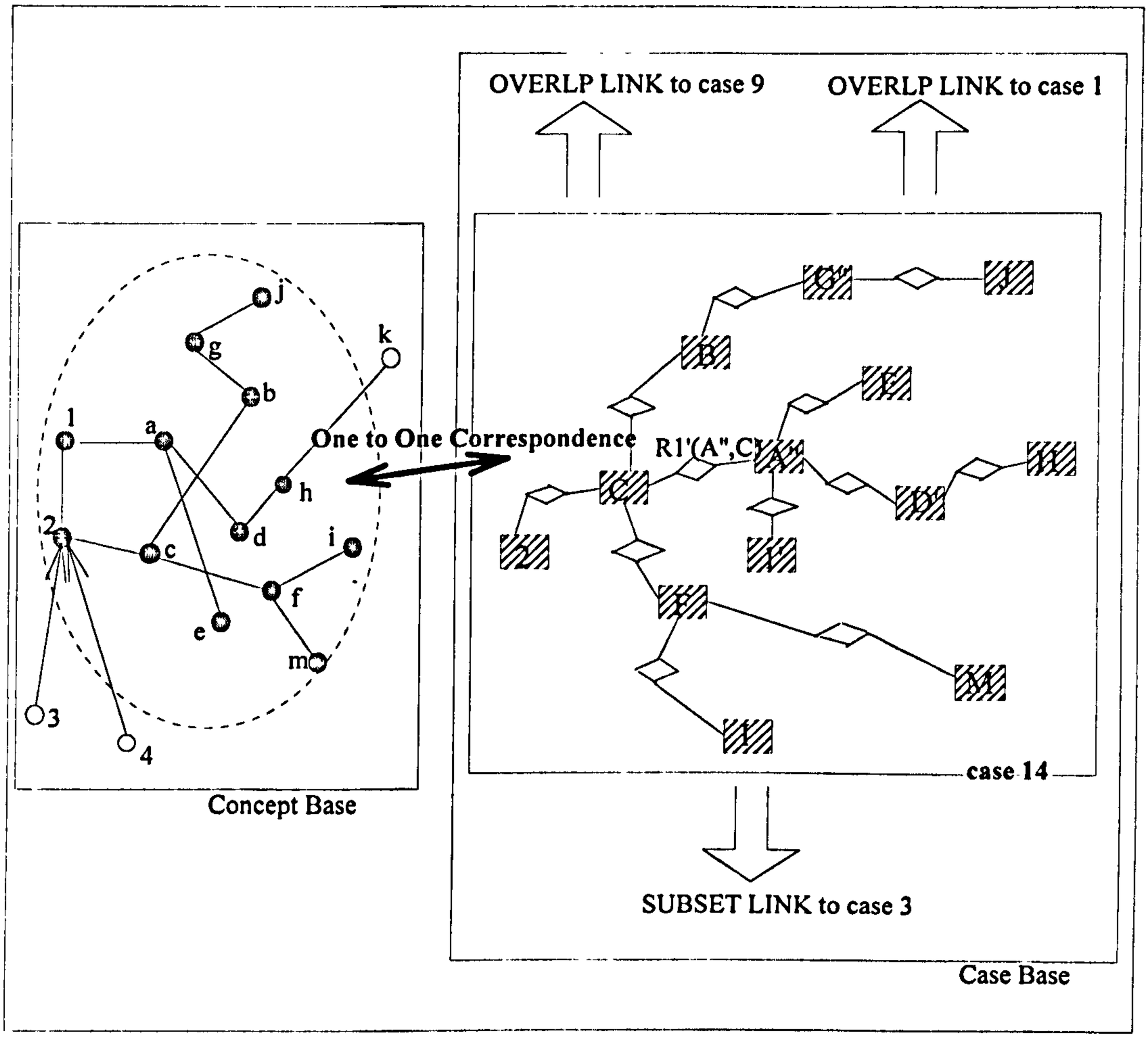


Figure 6.5 : The relevance of case 14 and its correspondence in the concept base

6.1.2.2 There is only the variation of the known entity (from the viewpoint of the DBKM) in the final conceptual data schema

In this situation there have been no effects on the structure of the concept base and the relations among system cases in the case base following the aforementioned design session. Only the exemplar link of the relevant known concept and its

variation will be built up for reference at the right time. In this example, under the meta rule G-Rule 7.4, the exemplar links of the variations of the non-case entities - A, G - are depicted in Figure 6.6.

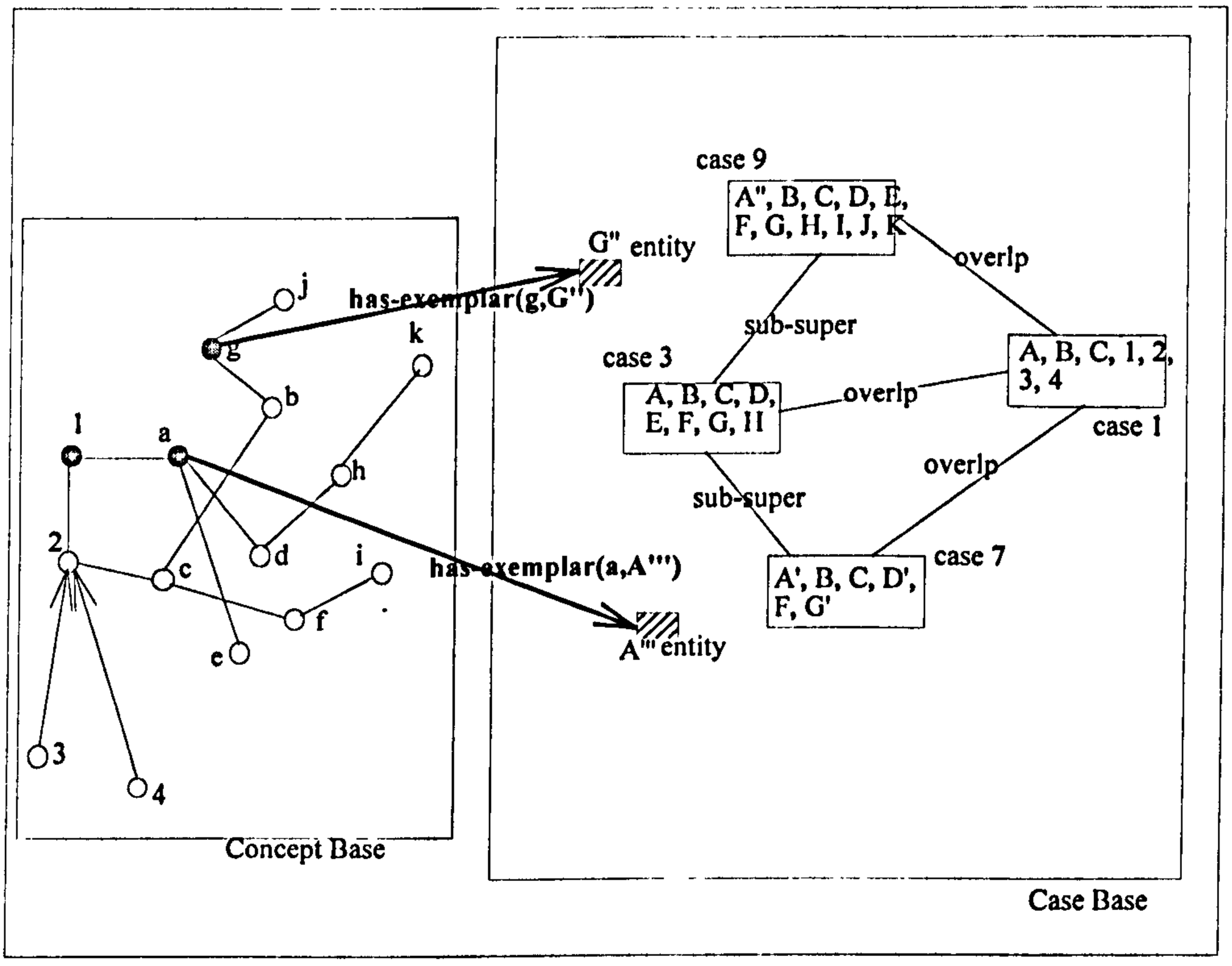


Figure 6.6 : The exemplar links of the variations of the non-case entities

is-typical

6.2 Discussion

6.2.1 The Knowledge-Based System View

In this respect, the distinctions between the DBKM and other frameworks of KBDDSs and CBR systems are as follows:

6.2.1.1 The distinctions between DBKM and the frameworks of other KBDDSs

The notion of methodological pluralism means that there is no single correct method of science, but there are many methods. The correct one is contingent on the specific problem to be studied, the kind of knowledge desired, etc. [Hirschheim, 1985]. Multimethodology refers to the whole area of utilising a plurality of methodologies or techniques within the practice of taking action in a problematic situation rather than being the name of a single methodology or a specific way of combining methodologies together [Minger and Gill, 1997].

1. Doing the right thing vs. doing the thing right

The roles of the DBKM and KBDDSs in database design are different. The constructs of the semantic data models and the theories of the logical data models are all based on the function of 'doing the thing right'. The KBDDSs discussed in

Chapter 2 encode the rules from these models and have been developed in order to promote the efficiency of the syntactical processes of the conceptual and logical database design. The object identification and interpretation stage of conceptual database design is concerned with how to identify and interpret the object concepts and relations among them. Its function is that of 'doing the right thing'. The DBKM which reuses previous concrete experiences is proposed here in order to enhance the effectiveness of this object identification and interpretation process.

2. The CBR paradigm vs. the RBR paradigm

The reasoning paradigm of the DBKM is different from the paradigm of the KBDDSs. The KBDDSs adopt the RBR paradigm which assumes that the nature (properties and function roles) of the objects in the reality is objectively fixed and unchangeable, and that the knowledge can be extracted and represented in the forms of rules. Thus, those KBDDSs, as long as the facts (the well-defined entities and relationships among them) are given, are very helpful for database design work. By completely ignoring the issue of how the relevant facts of the specific domain are acquired, the designer can always begin from scratch: every time the design session is initiated, the RBR agents in the current KBDDSs just refire the process (the inference rules).

Basically, the conceptual modelling work involves two contingent stages: the syntactical processing stage is contingent on the object identification and interpretation stage. In the object identification and interpretation stage, the

meanings of the objects are intersubjectively determined, not given. The intrinsic nature of the logical approach, which transcends all cultural and social factors, means that the RBR paradigm is not helpful in assisting the designers to identify and interpret the ill-defined objects and relations among them from the semantic rich reality to the well-defined entities and relationships among them in the specified UoD. In this research the DBKM adopts the CBR paradigm, which assumes that the reality exists as an undifferentiated background of unlimited possibilities and the knowledge is represented by nonanalytic large patterns - the cases. Thus, in the environment of the DBKM, every time the design session is initiated, the policy of overcoming the confusion of the object identification and interpretation of the conceptualisation work involves recalling the relevant products (the previous finished schemata stored as cases in the case base). In order to achieve this purpose, two complementary mechanisms, case retrieval and knowledge accumulation, are embedded into the framework of the DBKM. Through the case retrieval mechanism, the previous designs or part of them can be reused. Through the knowledge accumulation mechanism, the newly gained knowledge can be learned in an active manner.

6.2.1.2 The distinctions between DBKM and the frameworks of other CBR systems

The CBR paradigm is a research paradigm for addressing new problems by providing tools for investigation that go beyond what other research paradigms can provide rather than merely building intelligent or human-machine systems

[Kolodner, 1993]. As for AI in general, there are no universal CBR methods suitable for every domain of application. The challenge in CBR is to develop methods that are suited to problem-solving and learning in particular subject domains and to particular application environments [Malek and Labbi, 1995].

1. Two synergistic bases vs. one holistic base

The first method is to use a separated index framework - the concept base. This difference between the DBKM and other CBR frameworks, including CBDAs[60] (Case-Based Design Aids) architecture, such as ARCHIE-II [Domeshek and Kolodner, 1993], MIDAS [Domeshek et al, 1994] systems, and the CREEK framework reflects the *natural tendency* of the DBKM to consider the essence of the domain object-concepts in conceptual data modelling in the following two respects:

(1) The types and properties of relations of domain object-concepts.

The relations of domain object-concepts in those CBR frameworks mentioned above concern only the internal types. For example, the MIDAS uses the [part] relation to integrate the domain object-concepts of the chosen system such as *Hydraulics* (system) including *pumps*, *lines* and *fitting* objects, and the CREEK

---

[60] Case-based design aids (CBDAs) are computer systems based on two ideas: the importance of conceptual design and the usefulness of past concrete experiences during conceptual design. The intended purpose of the CBDAs is to improve the conceptual design process by playing the role of case repository and reminder [Domeshek and Kolodner, 1993; Domeshek et al, 1994a; Domeshek et al, 1994b].

architecture uses [part] and [sub-class (is-a)] relations to unify the object-concepts in diagnosing-problem domains such as in the car diagnosing domain: car *has-subclass* of vehicle and car *has-part* of wheel, fuel system, electrical system and engine[61]. Because, in those specific problem domains such as car problems (CREEK) and the conceptual design of specific artefacts domains such as building (ARCHIE-II) or aircraft (MIDAS) design, the properties of the internal relations of domain objects are very stable and can be pre-specified, the conceptual fundament that incorporates the domain object-concepts, problem-derived concepts and cases into a large semantic-based structure is sufficient and suitable. In conceptual data modelling, the concern of the relations between/among domain object-concepts are not only with the types (internal and external) but also with the properties (coverage properties for the internal type - generalisation, and degree, cardinality and attributes to the external type). For example, the Person in the university domain might be categorised into three total, overlapping sub-classes (is-a) such as teacher, student and staff, but might be divided into two total exclusive sub-classes, such as officer and soldier in the military personnel

---

[61] Here we just discuss the object-concepts of the problem domain of concern to those architectures. There are other kinds of concepts included in them. For example, in diagnosing the car domain of the CREEK, the goal-related concepts such as find-fault, diagnosis concepts and the functional-related fault concepts such as electrical-fault, fuel-system-fault, broken-carburettor-membrance concepts, as well as the object-concepts are all integrated as a whole conceptual fundament. In MIDAS, the hydraulic subsystem of the aircraft conceptual design, four types of issue-related concepts, such as (1) parameters: electric, pressure, weight, etc.; (2) pitfalls: contamination, cavitation, etc.; (3) Requirements (goals): safety, effectiveness, etc.; (4) Life cycle: design, repair, etc., as well as the object-concepts, are formed by the general index framework.

domain[62]. For handling the phenomena of multiple interpretations of the external relations between/among of the objects and of the generalisation relation between/among objects in the real world, the conceptual fundament that integrates the object-concepts and cases into one whole structure is inadequate and insufficient. So, the conceptual fundament of the proposed model is integrated by two separate, but corresponding structures. One is the concept base so that the domain object-concepts are just associated by two types of relations: [is-a] for internal relations and undefined relations for external relations[63]; and the other is the case base which store a number of concrete experiences.

---

[62] The discussion of the external relations of domain object-concepts can be seen in sections 2.1.1 and 2.1.2.

[63] The concept base is only formed by the domain object-concepts and their internal and external relations between/among them and does not integrate other concepts like those in CBDAs, PROTOS and CREEK - is suitable for this research for the following reasons:

(1) As a shared environment with the designers who are initiators and adapters in the conceptual design stage, the concept base of the DBKM is just used as an inference base and index framework to retrieve the relevant cases to assist designers in their conceptualisation work.

(2) The performance of the database cannot be checked directly by the finished conceptual schema, i.e. the entities and relationships between/among them. First, the finished schema will be transferred into a number of tables, and then each table will be normalised according to the theory of function dependence in the logical design stage (if the relational database management systems are used). Next, some normal-form tables might be denormalised, the indices will be created, and some triggers and stored procedures will be built in the physical design stage. Finally, the statistical table will be used, supported by the relational DBMS, to tune the indices, create other procedures or triggers in order to increase the performance of the data base. However, although the factors influencing the performance of the database are very complicated, the quality of the conceptual schema has been proved to be an important factor of database design. So, reusing the relevant finished conceptual schemata to assist designers is the main object of the DBKM.

(2) The functional role of the domain object-concepts.

The domain object-concepts in DBKM are not just taken to be peripheral and of second interest (i.e. the role of them is just for indexing) as they are in those CBR frameworks mentioned above. The roles of domain object-concepts are not only as indices (packed by their relations) to retrieve the relevant cases but also for co-ordinating the multiple interpretation of the objects in the real world. Therefore, the role of the concept base is not only as a base of index but also as a repository of the general domain knowledge.

2. General problem description vs. concrete problem description

The other method is to use the typical case of the specific UoD as a reference point to retrieve relevant cases without the concrete descriptions of the properties of domain objects. This difference between the DBKM and those CBR feature-based frameworks such as the PROTOS and CREEK frameworks that need concrete diagnosing symptom reflects the *pragmatic trend* of the DBKM to consider how to assist designers to do the right thing as their first step out of the confusion of the conceptual data modelling tasks. This consideration is based on the suggestions proposed by Egan and Schwartz [1979], and Storey and Goldstein [1993]. Egan and Schwartz [1979] postulated that experts can rapidly identify a conceptual chunking that relates many general concept clusters together and will systematically search drawings (50 electronic circuit drawings) by verifying local details of functional units in the long-term memory suggested by the general corresponding concepts in the conceptual chunking. Storey and Goldstein [1993]

described that human designers not only use their specialised knowledge obtained from previous work in the same application domain but also apply their general knowledge of the world. For example, when mentioning the word 'university' designers immediately suggest the need for entities such as Student, Course, Teacher, Department, and so on. This knowledge enables the designers to be active participants together with the end-users and encourages the end-users to communicate with the designers in the design process.

In this research, these notions proposed by the aforementioned researchers are synthesised and realised in the recalling process of the proposed DBKM. While designers initiate a specific UoD (first-pass focus), not only the typical case of the UoD is activated and the relevant entities (just the names of entities) in the activated typical case of the UoD are displayed immediately, but also the part of the concept base consisting of the corresponding general object concepts and relations is also activated[64] according to the relevance of the concept and case base[65]. So, these displayed entities can also be viewed as the general corresponding object-concepts of the concept case. The typical case and its corresponding part of the concept base work together as two reference points for retrieving other relevant cases in order to ease the burdens of the designers and

---

[64] This activation is by means of the exemplar links of object concepts in the concept base and entities in the typical case, and this mapping can be guaranteed by the relevance of the concept and case base discussed in section 4.2.3. After that, the relevant relations between/among the activated object concepts will also be activitated by referring to the relationships betwwen/among entities in the typical case.

[65] See section 4.2.3.

end-users. The only difference is that the typical case is in the foreground, but the corresponding part of the concept base is in the background.

6.2.2 The Knowledge Modelling View

Data modelling involves the design of a knowledge representation schema [Hirschheim et al., 1995]. From the viewpoint of the knowledge modelling, the distinctions between the DBKM, which uses the *case* knowledge-modelling discipline, and other knowledge models, which use the *rule* knowledge-modelling discipline, including current KBDDSs based on the beliefs underlying the fact-based school and those methods (e.g. the SAMPO and LEGOL/NORMA approaches) in the rule-based school, will be discussed.

6.2.2.1 Different beliefs between the DBKM and the knowledge models in the current KBDDSs

Current KBDDSs all use the context-independent rules, abstracted from the theories of database design (the semantic and logical data models) and generalised from the designers' experiences, as basic blocks to construct their knowledge-based database design aid environments. The essential prerequisite of using these environments is that the objects in the working UoD must be first identified and interpreted by their users. Accordingly, the confusion surrounding object identification and interpretation still exists.

The beliefs, which are inherited from the rationalistic tradition, underlying by these KBDDSs are the same as the fact-based data modelling approaches. In current KBDDSs, the problems of the conceptualising task are just pushed aside so that a clean (internal coherence and consistency) environment can be established. To moderate these beliefs, the proposed DBKM involves two synergistic bases, the concept and case bases, and in consequence the chaos of the conceptualising task can be eased to some degree. The DBKM is underpinned by the exemplar theory of object-concepts, which claims that the meaning of an object is represented by its exemplars in all possible contexts, and the case knowledge-modelling discipline (CBR), which accepts that the object-structure in a specified UoD is not of a unique and homogeneous status, but exists as different cases (possible worlds) in the case base. Under this environment, the rigid, strict logical beliefs of the current KBDDSs can be relaxed to handle the phenomena of multiple interpretations and polymorphic objects in the semantic-rich reality.

6.2.2.2 Different ideas concerning the DBKM and the knowledge models in the rule-based school

Although in the rule-based school, the knowledge-modelling technique - using the rules as media to develop approaches to facilitate the conceptual data modelling task - is the same as in the current KBDDSs, the consequences resulting from the different approaches they use are not the same. In the rule-based school, the main characteristic of these approaches, such as the LEGOL/NORMA approach originated by Stamper [Klein and Hirchheim, 1987; Hirschheim et al., 1995] and

the SAMPO method [Auramäki et al., 1988; 1992a,b] associated with the Language Action (LA) view, is that they are domain *dependent*, but the environments of the current KBDDSs are all domain *independent* (see Figure 2.2 - Classification of the current KBDDSs - in section 2.1.1). Wittgenstein's [1958] notion of the rule-governed character of language use has led scholars in the fact-based school enthusiastically to seek the *general templates (rules)* of *language use* (especially in the semantic and pragmatic aspects) in some particular domain, such as business or the office, in order to handle the task of data modelling in information system development. However, the illusions of the LEGOL/NORMA method in tackling different interpretations and of the SAMPO approach in handling data modelling, as well as the weaknesses of their (LEGOL/NORMA and SAMPO) constructs make them less applicable and less useful for practising database designers [Hirschheim et al., 1995]. For example, the rules of human communication based on the speech act theory in the SAMPO office modelling approach have led it to be recognised as a basis for developing group-oriented communication tools rather than as a data modelling approach. In addition, although the philosophical root of the LEGOL/NORMA approach is hermeneutics, the important perspective of preunderstanding, which is a necessary condition of being able to interpret anything in the world, has not been fulfilled in the LEGOL/NORMA environment. Consequently, the LEGOL/NORMA is just regarded as a special kind of semantic model which provides rich semantic primitives and rules to model the specific UoD. In other words, the facility of helping the designers to expand their horizons to understand the meaning of a specific text is not supported in the LEGOL/NORMA environment.

The meaning of an object is that it is not given 'out there', but depends on its intended use, thus resulting in multiple interpretations and polymorphic objects. The current methods in the rule-based school seek to provide a unified interpretation based on the formal or pre-defined syntax rules (according to the semantic and pragmatic aspects of the language of user communities) to control the problems of interpretation in certain domains. The present research turns towards another notion which differs from the idea adopted by the rule-based school. The notion of proposing the DBKM is based on the view of the 'case-governed character of expert behaviours', but not on the view of the 'rule-governed character of language use'. This results in using the case knowledge-modelling discipline to construct a knowledge model to manage the interpretation problems in which the multiple interpretations and polymorphic objects can be handled more easily. In the case knowledge-modelling discipline, the case is a basic unit of the knowledge which provides a ballpark solution to support the problem-solving process. In this research, the case involves the finished schema of a specific UoD which is an interpretation of the specified UoD. The case base consists of a number of inter-related cases which might be from the same UoD or different UoDs. To help the conceptualising task, the relevant cases can be retrieved so that designers can pre-understand the problem domain and ease the communication with users.

6.2.3 The Limitations of Current DBKM

6.2.3.1 The static aspect of the UoD

Conceptual modelling is a way of thinking about the UoD using models organised around real-world concepts [Rumbaugh et al, 1991; Loucopoulos and Zicari, 1992; Boman et al, 1997]. According to the modelling orientation, there are three orthogonal kinds of models that concentrate the UoD into different dimensions: the conceptual data model, dynamic model and functional model. The conceptual data model, represented graphically with semantic data diagrams, describes the static aspect of the UoD in terms of entities and relationships. The dynamic model, represented graphically with state diagrams,describes the control aspect of the UoD in terms of the events and states. The functional model, represented graphically with data flow diagrams, describes the computational aspect of the UoD in terms of the values and processes.

However, the conceptual data model provides the essential framework into which the dynamic and functional models can be placed. Changes and transformations are meaningless unless there is something to be changed or transformed. Objects are the units into which we divide the world, the molecules of the UoD with which the system is concerned. It is only when the inherent object concepts of the UoD are identified, interpreted, organised, and understood that the details of data structures and functions can be addressed effectively. Therefore, although the

theory of the DBKM in this phase is limited to the static part of the UoD[66], which seeks to assist the designer to carry out the conceptual data modelling work, it can be used as a fundamental base to further model the dynamic and functional aspects of the UoD.

6.2.3.2 The passive generalising of the domain cases and their corresponding concepts and relations

In order to provide a source of analogical thinking for the designer, the case base of the DBKM includes the domain cases as well as the system cases. The *domain cases* are of the abstract structures which are generalised by the *system cases* in the same domain: for example the transportation control domain might include train and airport control systems. In this research, the mechanism of knowledge accumulation in the DBKM framework is just for the system cases. In other words, for the DBKM agent, the newly gained *system cases* will be retained in the case base by its active learning mechanism. However, the *domain cases* can still be generalised and stored, but by the passive method: through the knowledge manager and the knowledge engineer.

No matter whether domain cases or systems cases are involved, the meanings of the objects and relations in the cases are interpreted and provided by humans. Humans and computers have their own merits. In order to exploit their respective talents, Norman [1997: 112] states, 'it should possible to develop a symbiotic,

---

[66] Those KBDDSs discussed in Chapter 2 are also dedicated to this aspect of the UoD.

complementary strategy for co-operative interaction'. In the DBKM environment, the people are object providers and semantic interpretators who are responsible for the interpretation of the UoD based on its background culture and conventions, and the computers are rich external sources of information. Therefore, this limitation of the DBKM is not a negative one, but a positive consequence of bridging the gap between humans and computers.

Chapter 7

Conclusion and Future Work

The previous chapter provided a comprehensive view for understanding the function of the constructed DBKM, discussed the distinctions between the DBKM architecture and the frameworks of current KBDDSs and CBR systems (from the knowledge-based system view) and the distinctions between the DBKM between those models in the current KBDDSs and rule-based data modelling approaches (from the knowledge-modelling view), and evaluated the limitations of the current DBKM. This chapter highlights the main research contributions and suggests appropriate directions for future research work.

7. 1 Conclusion

The quality of conceptual data modelling has been recognised as a critical factor of database design. Although the KBDDSs have been developed to try to automate this modelling process, the confusion of current conceptualising work from the semantic-rich reality to the specific UoD is still not resolved. In this dissertation, a theoretical architecture - the DBKM - is proposed to tackle this situation. The contributions of this study may be placed into two categories: (1)

for the knowledge-based database design aid environment; (2) for the constructed DBKM itself.

7.1.1 For the Knowledge-Based Database Design Aid Environment

The proposed DBKM is a theoretical architecture which provides a valid basis for complementing the environments supported by the current KBDDSs and a rational basis for creating the symbiosis of humans and computers so that the confusion of the current conceptualising task can be alleviated.

7.1.1.1 Constructing a theoretical model

The DBKM adopts the theories of conceptual knowledge from the cognitive school, including (1) the notions of the relevance of the mental world and mental model (scene-schema mental model), which are used as premises to construct the theory of dual-base which include two different kinds, but complementing each other synergistically, of knowledge structures; (2) the exemplar view of the object concepts, which is used to handle the multiple interpretations and polymorphic objects; and (3) the CBR paradigm, which approximates human behaviour very closely during the problem-solving process to alleviate the difficulties of object identification and interpretation in conceptual data modelling by reusing the relevant concrete cases. Except for the above underlying theories of the dual-base model, to further ground the proposed dual-base model, the notions of relation and

function from set theory are used to provide valid bases of help in reusing the relevant old cases and facilitating the acquisition of new cases, and of interplay of these two bases.

7.1.1.2 Analysing an elaborated process model

Two different issues, design by reuse and design for reuse, are involved in the reuse of concrete experiences. In essence, they are two sides of the same coin. Two mechanisms, the case retrieval and knowledge accumulation mechanisms, are derived from the theory of DBKM and are shown to address these implications. By means of the case retrieval mechanism, the relevant cases can always be retrieved at the right time from the very large memory pool, no matter how large and complex it is; and by means of the knowledge accumulation mechanism, the newly gained knowledge can be allocated in an appropriate place in the knowledge repository to be recalled in the next design session.

7.1.1.3 Providing an intelligent component

Intelligence is a matter of doing the right thing. In post-modern AI, the purpose of AI has been shrunk from that of 'ultra-intelligent machines - surpassing human intellect' and 'intelligent agents - imitating the human intellect' to 'intelligent components - assisting people in certain respects', i.e. applying AI technology in limited ways to help people interact with computer systems. Thus, the implication

of this trend in the focus of current AI is to develop systems that are not stupid, rather than systems that are intelligent [Chandrasekaran, 1994; Riesbeck, 1996; Shyrme, 1997]. The lever of this thought is to create an environment of purposeful co-operation between humans and computers. There are circumstances when it is pointless to try to be more robust since, without a theory and mechanism on which co-operation can be based, the symbiosis of people and machines will be sought in vain. The theory and mechanisms of the DBKM are proposed here to facilitate object identification and interpretation in the conceptual data modelling as an example of this leverage.

7.1.2 For the Constructed DBKM Itself

The construction of current KBDDSs is influenced by the beliefs underlying the fact-based data modelling school, which assumes that developers and users of the database can construct a clear mapping from the semantic-rich reality to a specific UoD without any hesitation and confusion, and in consequence logical approaches (propositional calculus and first-order predicate logic) are adopted as the main patterns to develop the KBDDSs. Therefore the generalised, abstracted 'Rule' becomes the basic unit of knowledge in these KBDDSs. In this study, the conceptual knowledge fundament of DBKM involves two kinds of knowledge structure - the concept base consisting of the object-concepts and relations, and the case base including the interrelated cases. A knowledge-based agent (KBDDS), based on this proposed conceptual knowledge fundament, can make its knowledge

meaningful and non-subjective, moderate the beliefs underlying the fact-based school, and provide a hermeneutic environment to ease the chaos of the conceptualising task.

7.1.2.1 Making the knowledge meaningful and non-subjective

Using the RBR paradigm makes the knowledge of current KBDDSs meaningless and empty. By contrast a knowledge model, the DBKM has the following basic characteristics that make its knowledge meaningful and non-subjective:

I. Concepts are meaningful that are directly tied to the experience base (case base). This avoids emptiness.

II. Cases are constant experiences that mostly involve successful functioning, and stringent UoD constraints based on the culture and conventions of the specified UoD. This avoids subjectivism.

7.1.2.2 Improving theoretically the beliefs underlying the fact-based school

Using the RBR paradigm means that current KBDDSs are useful only when the objects (relations) have been identified and interpreted. The potential dilemmas of object identification and interpretation still exist because the beliefs underlying the KBDDS environments which are the same as those of the fact-based school, remain unchanged. In this study a knowledge model (DBKM) is constructed to

relax the frame of reference of current KBDDSs and the fact-based school, but in its own way.

I. The moderated objective belief

(I) The anomaly correspondence (contrasting the one-to-one correspondence): The corresponding anomaly can be co-ordinated by the DBKM. For example, the anomaly correspondences between 'marks on paper' (descriptions of sentence) and states of affairs (SoA) in the real world can be handled by the stored cases (possible world). The ambiguous meaning of the word 'sales' in different groups such as the sales person, the accountant and the lawyers, can be tolerated by the exemplar view of object concepts. By means of the relevance of the two bases - the concept and case base - the relevant cases can all be retrieved as potential solutions to assist people to interpret the meanings of objects and SoA in the real world.

(II) The non-fixed and non-immutable entity-structure (contrasting the fixed and immutable entity-structure): The non-objective entity-structure and unbound boundaries which are dependent upon the practical problems of human activity are accepted by the DBKM. In the DBKM, the different bounded entity-structures, which might be from the same UoD or different UoDs, are stored as cases in the case base. The relevant cases can be recalled in designing a conceptual schema in the specific UoD in order to alleviate the designers' difficulties of object identification and interpretation.

(III) The non-excluded middle (contrasting the principle of excluded middle): The principle of the non-excluded middle is allowed by the DBKM. In the same UoD the content and role of an object in one conceptual schema might be

different or might not even exist in other conceptual schemata which are dependent upon the conventions of the organisation or groups. In the DBKM, this violence of the excluded middle principle is permitted so that the multiple interpretations can be managed.

(IV) Context dependence (contrasting the context independence resulting from the language neutrality): Context dependence can be provided by the DBKM. Basically, the cases in the case base are finished schemata in some contexts, but not abstracted, generalised rules, which can be used to offset the language neutrality in the fact-based data modelling.

II. The moderated descriptive belief

The descriptive belief is relaxed in the following two ways:

(I) The example view of concept is the meaning of a concept as represented by its exemplars, i.e. the essence of concept (intension) is however the denotation (extension) of concept instances at all times in all possible worlds.

(II) The mechanism of evoked typical case in a specified UoD behaves as a deontic or epistemic operator to allow some objects (relations) must be in a specified UoD (modelling the notions of belief).

III. The moderated consensus belief

The cases, which might be in the same UoD or from different UoDs, in the case base can be viewed as a federation of conceptual schemata where each local schema is only locally consistent but at the global level does not assume any consistency or consensus.

7.1.2.3 Providing a hermeneutic environment

The DBKM is constructed by the exemplar view of the object concept and the CBR paradigm, and provides a hermeneutic environment to ease the chaos of the conceptualising task to some degree.

The exemplar view of object concepts denies that the essence of an object concept (intension) is defined by a set of the defining features that are singly necessary and jointly sufficient, but is the denotation (extension) of concept instances in all possible contexts. The CBR paradigm assumes that the world exists as an undifferentiated background of unlimited possibilities. The entities and relationships which designers discover in the background are not a matter of observation, but of interpretation. In the CBR paradigm, the conceptual database design begins the object identification when the designer discovers some aspect of the background upon which it is useful to focus. This discovery is called a 'breaking down' [Winograd and Flores, 1987; Patula, 1992] through which separable objects are identified out of the undifferentiated background and become 'present-at-hand'. Once the interpretation of the objects (relations) has been made, the present-at-hand entities are used to inform a design using either a rule-based or case-based process. Of course, new discoveries and interpretations can be made at any time, which would result in new entities 'breaking down' and becoming 'ready at hand'.

Therefore, from the viewpoint of hermeneutics, the proposed knowledge model rests on the following two positions: it seeks (1) to support a source of preunderstanding for the designer which gives narrow structures to designers' horizon; (2) to provide a space of potential breakdowns which provides an environment to help anticipate and cope with these breakdowns.

7.2 Future Research Work

7.2.1 Theoretical Issues of the DBKM Itself

The conceptual modelling of the UoD includes three complementary models: the conceptual data model, dynamic model and the functional model. All three models are necessary for a full understanding of a problem, although the balance of importance among the models varies according to the particular kind of application. In order to support the conceptual modelling in a whole environment, the DBKM will be extended to accommodate the dynamic and functional aspects of the UoD. The dynamic model shows the behaviour on the objects and specifies the allowable scenarios that may occur in the specified UoD in terms of events and states. The functional model shows the operations on the objects and specifies the possible functional paths that may be indicated in the specified UoD in terms of the processes and values (the data flows). The extension of the dynamic aspect of the UoD depends on further research into the nature of the *event-type concepts* in

conceptual modelling. The extension of the functional aspect of the UoD depends on research into the nature of the *process-type concepts* in conceptual modelling.

7.2.2 Availability Issues of the DBKM Environment

As a knowledge-level model which abstracts from implementation-related considerations to focus on the theoretical grounds by providing the right level of description of a problem solveing agent [Newell, 1982; Schreiber et al., 1990; Schreiber, 1992; Motta, 1997], the DBKM is not directly linked with the actual implementation and should not contain all information necessary for building the implementation. In the process of implementing a system, it will always be necessary to add specific information. The implementing process is constrained by the specification of the DBKM, but a large number of implementation issues are not relevant to the DBKM. Therefore, the aim of the DBKM constructed in this research is to provide the appropriate specification for implementing a knowledge-based system for reusing the conceptual data schemata. However, in order to enhance the availability of the DBKM environment, the following issues, based on the essence of the DBKM, should be considered: the layered user interface, the parallel computing algorithm and the client/server architecture aspects - all of which have implications for future research directions.

(1) The layered user interface

To reduce the complexity of the conceptual modelling task, it may be advantageous to design computerised modelling aids that support the conceptual design work at various levels of abstraction and can facilitate transitions among those levels [Srinivasan and Te'eni, 1995]. By means of this layered user interface, not only is *readability* improved by the possibility of displaying the contents of a conceptual schema from a general view to a detailed description, but the *reusability* of a conceptual schema can be achieved conveniently by providing the possibility of reusing part of a conceptual schema.

Based on the process model of the DBKM, an example of the layered user interface might be drafted as follows as shown in Figures 7.1, 7.2 and 7.3. The main content of a case is divided into three layers in the order of increasing complexity: (1) the Entity-Attributes layer (EA layer) concerns the attributes of entities and can be used in the adaptation of the attributes of an entity, as shown in Figure 7.1; (2) the Entity-Relationship Layer (ER-layer) concerns the relationship of the related entities and can be used in the adaptation of the property of the relationship of the related entities, as shown in Figure 7.2; and (3) the whole structure layer can be used to provide a holistic comprehension of the problem doamin, as shown in Figure 7.3. However, the appropriate interfacing mechanism needs to be further examined by referring to the techniques of graphic user-interface design.

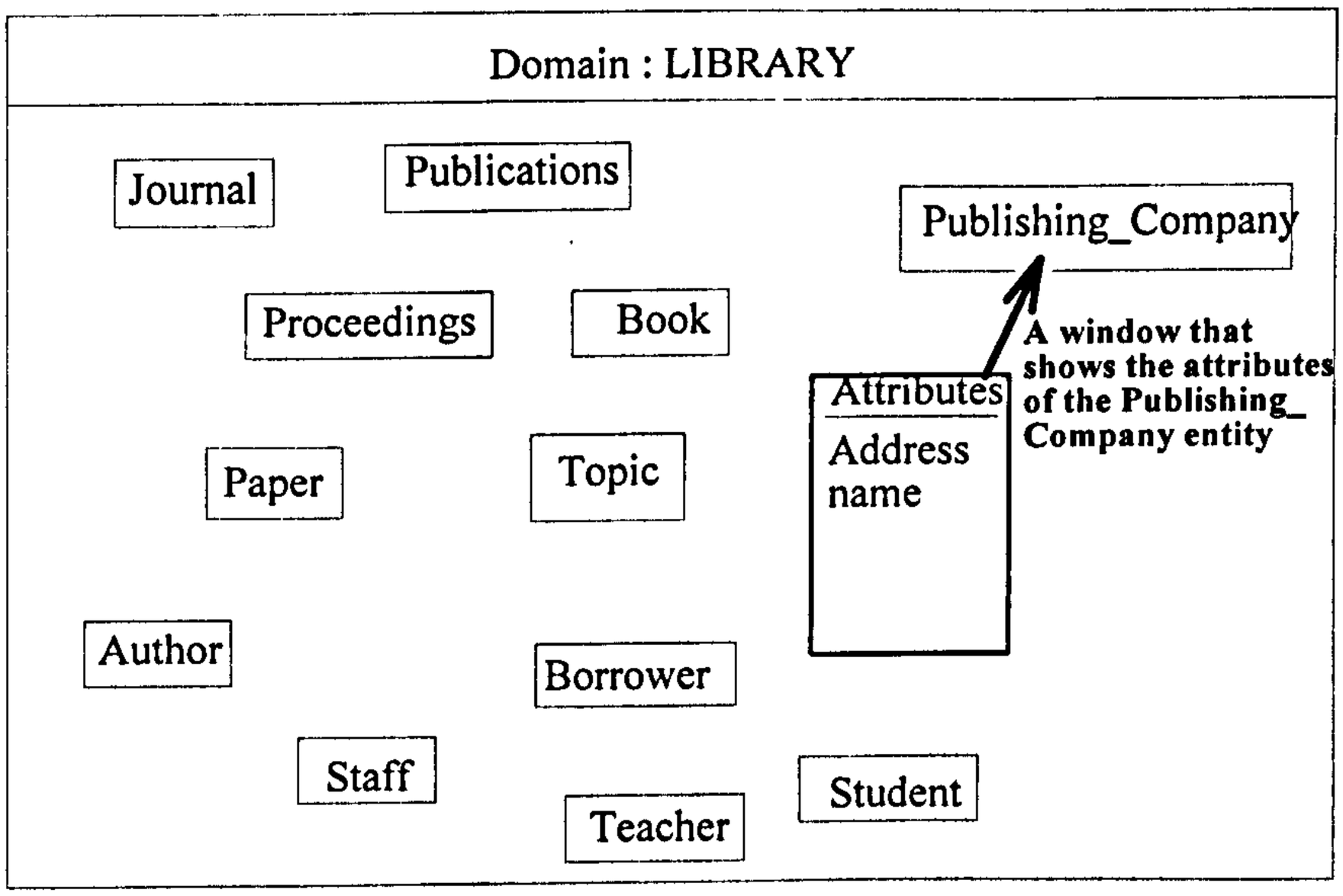


Figure 7.1 : The first layer of the case structure: The Entity-Attributes layer

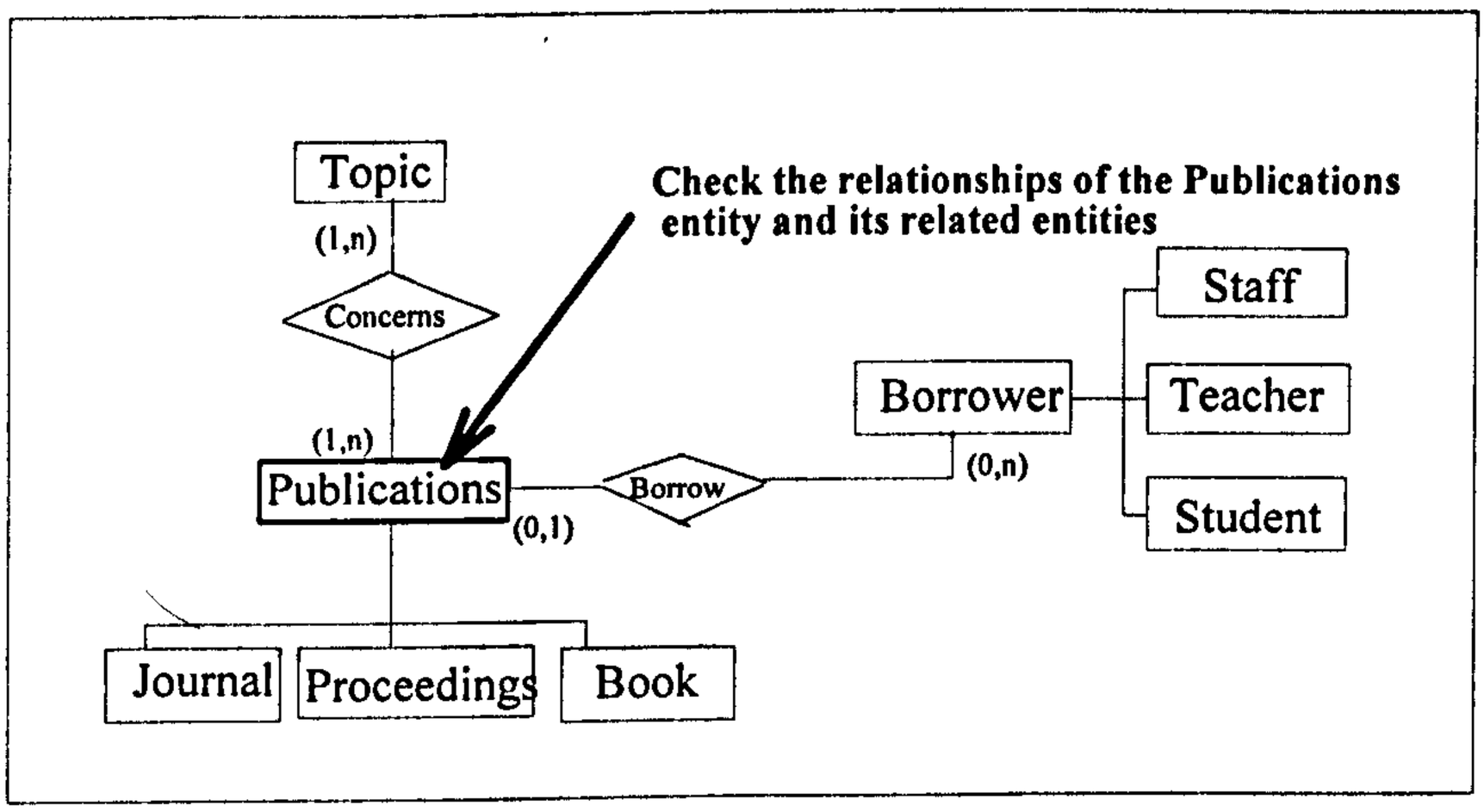


Figure 7.2 : The second layer of the case structure: The Entity-Relationship layer

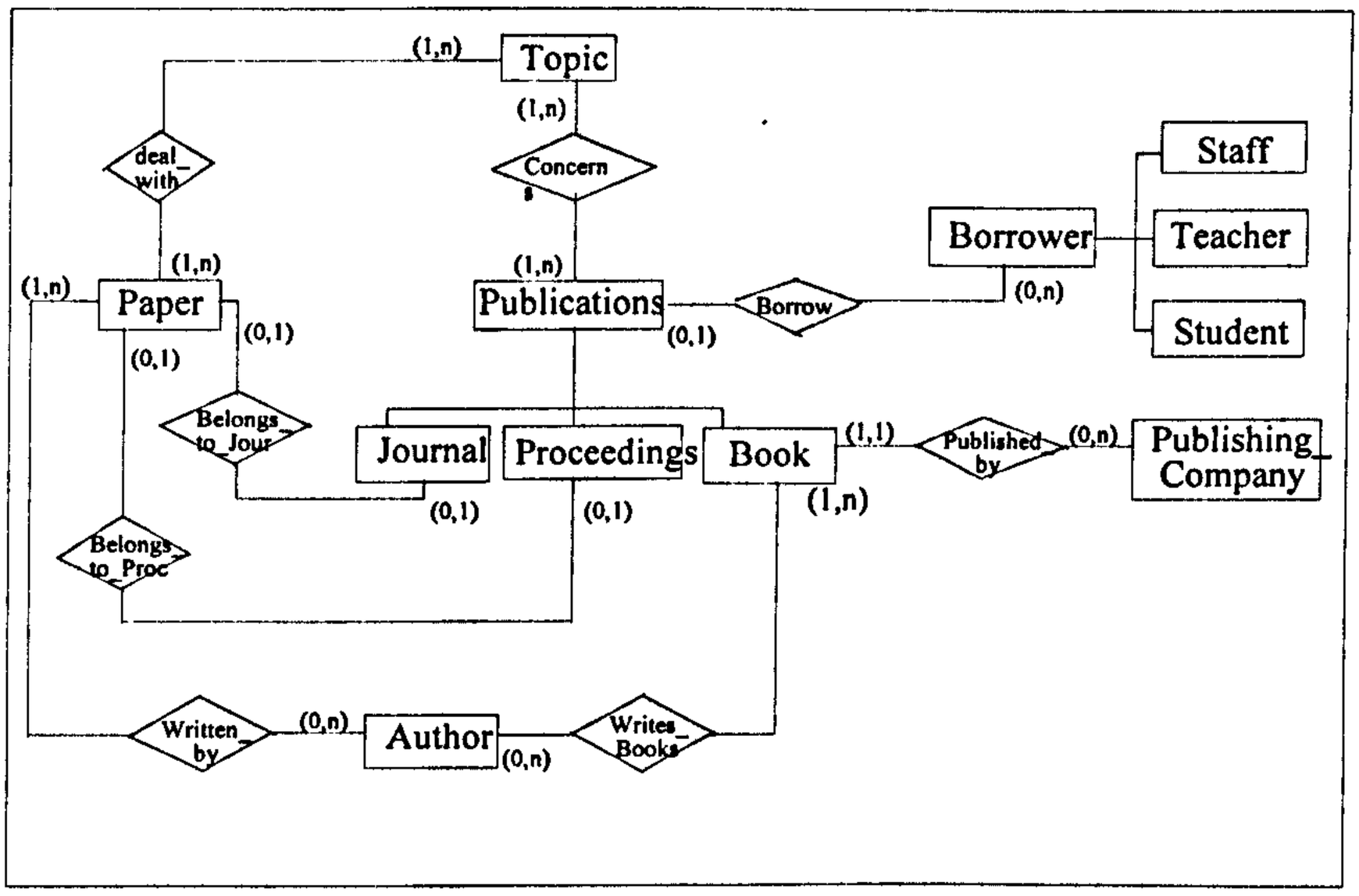
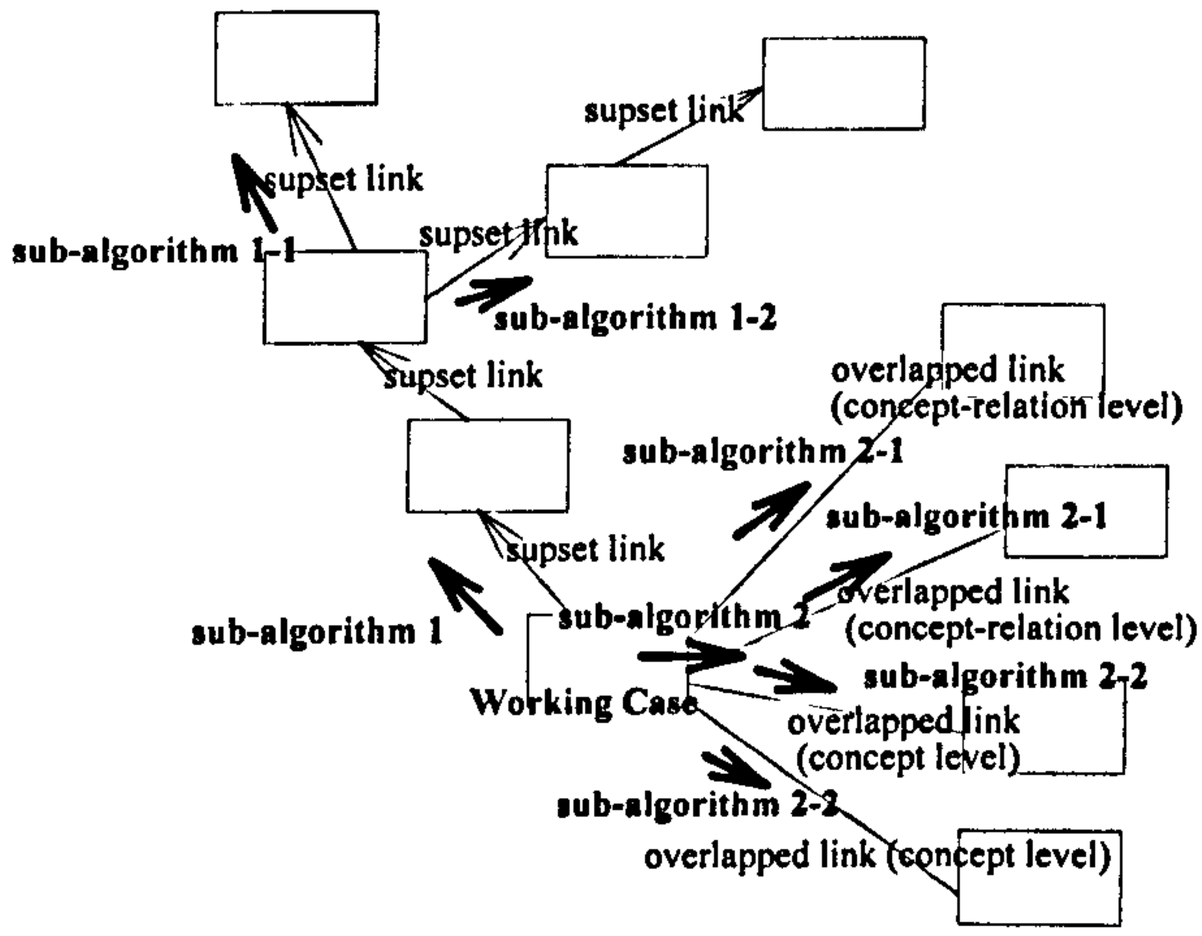


Figure 7.3 : The third layer of the case structure: The Whole Structure layer

(2) Parallel computing algorithm

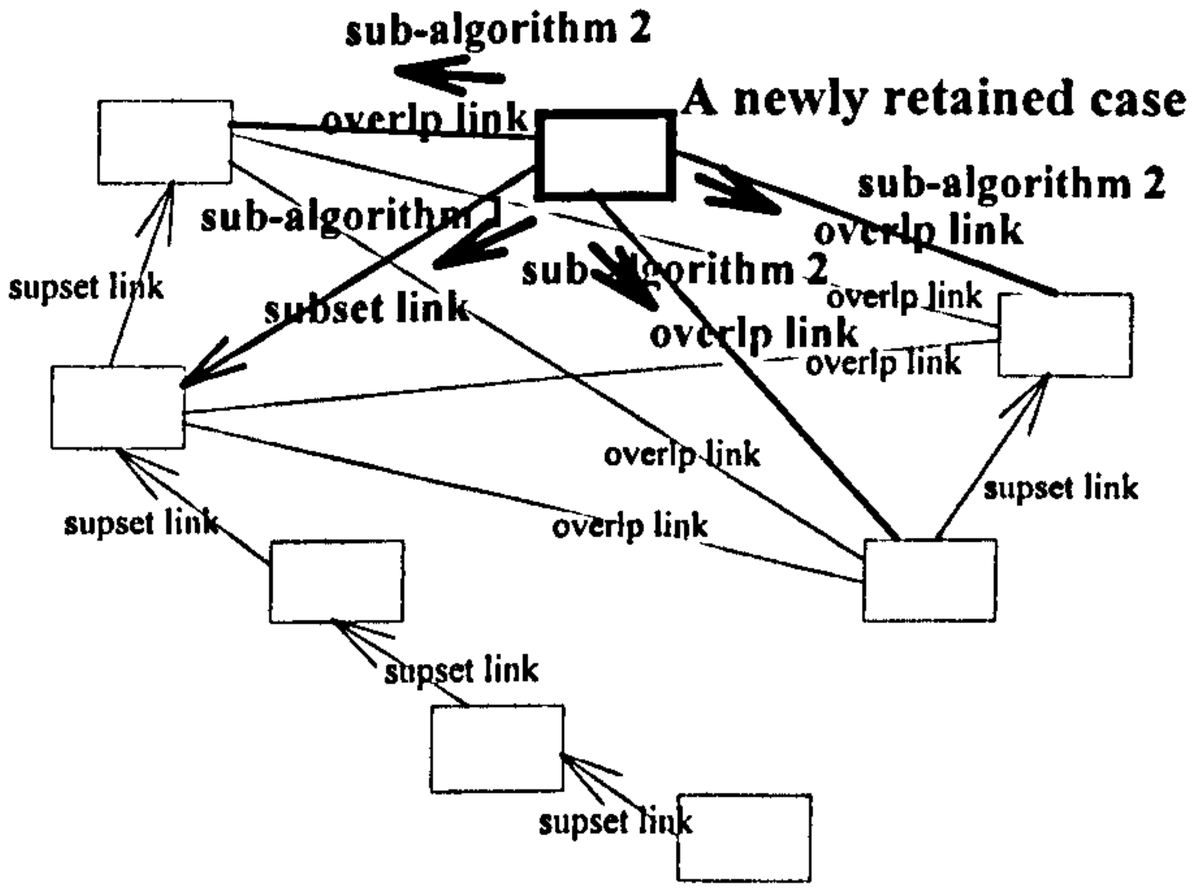
Two independent relations (the sub-super and overlp relations) are inherent in the system cases. In order to locate the relevant cases efficiently, the parallel algorithm technology can be used to implement the case retrieval (the find sub-process) and knowledge accumulation mechanisms (the new-case-retained process) of the DBKM. For example, because of the property of the finding element, the finding algorithm can be designed as two or more parallel sub-algorithms to find the relevent cases which are either of the supset cases or the overlapped cases of the working case, as shown in Figure 7.4. For the new-case-retained process, based on the independent property of the relations of a case, the contents of the supset, overlapped and subset links of the located relevant cases and the newly retained case might be established by three parallel sub-algorithms, as shown in Figure 7.5 (including only the subset and overlapped link

algorithms). However, the appropriate parallel-computing mechanism needs to be further investigated by referring to the techniques of parallel algorithm.



1. There are two parallel sub-algorithms:
   (1) sub-algorithm 1 (for the superset cases of the working case);
   (2) sub-algorithm 2 (fro the overlapped cases of the working case)
2. The sub-algorithm 1 might be forked into
   two parallel sub-algorithms:
   (1) sub-algorithm 1-1;
   (2) sub-algorithm 1-2.
2. The sub-algorithm 2 might be forked into
   two parallel sub-algorithms:
   (1) sub-algorithm 2-1(for the concept-relation level overlapped case);
   (2) sub-algorithm 2-2 (for the concept level overlapped case).

Figure 7.4 : An example of the parallel algorithms of the find sub-process



There are two parallel sub-algorithms:
(1) sub-algorithm 1 (for the subset link);
(2) sub-algorithm 2 (for the overlapped link)

Figure 7.5 : An example of the parallel algorithms of the retained case process

(3) The client/server architecture

By means of combining Information Technology (IT) infrastructures including the gateways, networks, front-end tools, and knowledge bases, the DBKM architecture can be developed as a knowledge server, whose primary function is that of a retrieval-only server used for conceptual data schema design support. In order to create schemata-sharing surroundings and effectively manage knowledge-modelling engineering, client/server computing, which involves a set of technologies such as communications, transaction processing, network management, network, middleware, security and so on, can be adopted to support this environment. For example, the simple framework of client/server processing environment of the DBKM might be depicted as in Figure 7.6. The clients can access the information stored in the knowledge server by different paths. However, the appropriate schemata-sharing environment needs to be further considered by the current methods or tools of the client/server architecture.

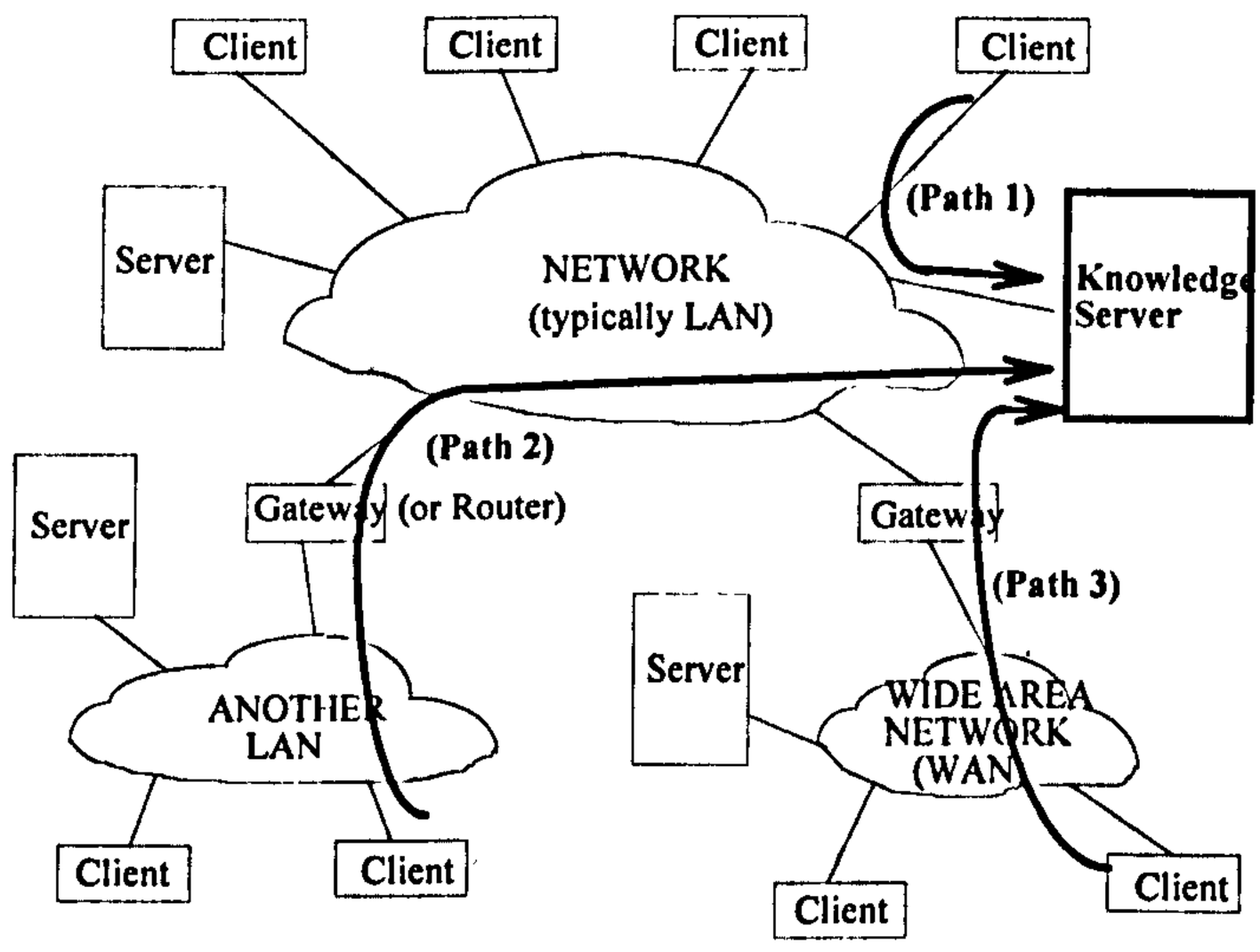


Figure 7.6 : An example of a simple framework of the client/server processing environment of the DBKM

References

Aamodt, A. (1989), Towards Expert Systems that Learn from Experience. In *Proceedings of a Workshop on Case-Based Reasoning*, Pensacola Beach Florida, 181-187.

Aamodt, A. (1990a), A Computational Model of Knowledge-Intensive Learning and Problem Solving. In Wielinga, B. et al. (eds.), *Current Trends in Knowledge Acquisition*, IOS Press, Amsterdam, 1-20.

Aamodt, A. (1990b), Knowledge-Intensive Case-Based Reasoning and Sustained Learning. In *Proceedings of the 9th European Conference on Artificial Intelligence ECIA-90*, 1-6.

Aamodt, A. (1994a), Explanation-Driven Case-Based Reasoning. In Wess, S., Althoff, K-D. and Richter, M. M. (eds.), *Topics in Cased-Based Reasoning*. Springer-Verlag, Berlin Heidelberg, 274-288.

Aamodt, A. (1994b), A Knowledge Representation System for Integration of General and Case-Specific Knowledge. In *Proceedings of IEEE Conference on Tools for Artificial Intelligence*, 1-5.

Aamodt, A. and Nygard, M. (1995), Different roles and mutual dependencies of data, information, and knowledge - an AI perspective on their integration. *Data&Knowledge Engineering* 16, 191-222.

Adelson, B. and Soloway, E. (1985), The Role of Domain Experience in Software Design. *IEEE Transactions on Software Engineering* SE-11 (11), 1351-1360.

Anderson, J. R. (1995), *Cognitive Psychology and its Implication* (4th edition). Freeman and Company, New York.

Auramäki, E., Hirschheim, R. and Lyytinen, K. (1992a), Modelling Offices through Discourse Analysis: The SAMPO Approach. *The Computer Journal* 35 (4), 342-352.

Auramäki, E., Hirschheim, R. and Lyytinen, K. (1992b), Modelling Offices through Discourse Analysis: a Comparison and Evaluation of SAMPO with OSSAD and ICN. *The Computer Journal* 35 (5), 492-500.

Auramäki, E., Lehtinen, E. and Lyytinen, K. (1988), A Speech-act Based Office Modeling Approach. *ACM Transactions on Office Information Systems* 6 (2), 126-152.

Bareiss, R.(1989), *Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning*. Academic Press, San Diego.

Barletta, R. (1994), A Hybrid Indexing and Retrieval Strategy for Advisory CBR Systems Built with REMIND. In *Proceedings of the Second European Workshop on Case-Based Reasoning*, Chantilly France, EWCBR, 49-58.

Barletta, R. and Hennessy, D. (1989), Case Adaptation in Autoclave Layout Design. In Hammond, K. (ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, San Francisco, Morgan Kaufmann, 203-207.

Bartlett, F. C. (1932), *Remembering: A study in experimental and social psychology*. Cambridge University Press, Cambridge.

Batini, C., Ceri, S. and Navathe, S. B. (1992), *Conceptual Database Design: An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Redwood.

Batra, D. and Davis, J. G. (1992), Conceptual data modelling in database design: similarities and differences between expert and novice designers. *Int. J. Man-Machine Studies* 37, 83-101.

Batra, D. and Zanakis, S. H. (1994), A conceptual database design approach based on rules and heuristics. *European Journal of Information Systems* 3 (3), 228-239.

Beynon-Davies, P. (1992), The realities of database design: an essay on the sociology, semiology and pedagogy of database work. *Journal of Information Systems* 2, 207-220.

Birnbaum, L. (1991), Rigor mortis: a response to Nilsson's Logic and artificial intelligence. *Artificial Intelligence* 47, 57-77.

Boman, M., Bubenko, Janis A., Johannesson, P. and Wangler, B. (1997), *Conceptual Modelling*. Prentice Hall, London.

Bouzeghoub, M., Gardarin, G. and Metais, E. (1985), Database Design Tools: An Expert System Approach. In *Proceedings of the 11th International Conference on Very Large Database*, Stockholm, 82-95.

Bower, G. H., Black, J. B. and Turner, T. J. (1979), Scripts in memory for text. *Cognitive Psychology* 11, 177-220.

Brewer, W. F. and Treyens, J. C. (1981), Role of schemata in memory for place. *Cognitive Psychology* 13, 207-230.

Brooks, L. (1978), Non-analytic concept formation and memory for instances. In Rosch, E. and Lloyd, B. B. (eds.), *Cognition and Categorization*. Lawrence Erlbaum Associates, Hillsdale, 169-211.

Castano, S. and Pernici, B. (1996), Reuse at the Conceptual Design Level: Models, Methods, and Tools. In *Proceedings of the tutorials of 15th International Conference on Conceptual Modeling*, Cottbus Germany, Oct. 7-10, 104-157.

Chandrasekaran, B. (1994), AI, Knowledge, and the Quest for Smart Systems. *IEEE Expert* 9 (6), 2-5.

Chang, T. Y., Hurrion, R. D. and Wu, C. H. (1997), A Theory of Dual-Base Knowledge Model - integrating the concept and case base for conceptual data schemata reuse. In *Proceedings of the 5th European Conference on Information Systems*, Cork-Ireland, June 19-21, 1010-1026.

Chang, T. Y., Wu, C. H. and Hurrion, R. D. (1996), A Concept-Case Knowledge Model for Conceptual Database Design. In *Proceedings of the Challenges Workshops of 15th International Conference on Conceptual Modeling*, Cottbus Germany, Oct. 7-10, 22-35.

Chase, W. G. and Simon, H. A. (1972), The mind' eye in Chess. In *Proceedings of the Eighth Annual Carnegie Symposium on Cognition*, Carnegie-Mellon University, Pittsburgh, Pennsylvania, May 19.

Chase, W. G. and Simon, H. A. (1973), Perception of chess. *Cognitive Psychology* 4, 55-81.

Chen, P. P. (1976), The entity-relationship model: toward a unified view of data. *ACM transactions on Database Systems* 1 (1), 9-36.

Cherniak, C. (1984), Prototypicality and deductive reasoning. *Journal of Verbal Learning and Verbal Behavior* 23, 625-642.

Chi, M. T. H., Feltovich, P. J. and Glaser, R. (1981), Categorization and Representation of Physics Problems by Experts and Novices. *Cognitive Science* 5, 121-152.

Choobineh, J., Konsynski, B. R., Mannino, M. V. and Nunamaker, J. F. (1988), An expert system based on forms. *IEEE Software Engineering*, February, 242-253.

Clancey, W. J. (1991), The Frame of Reference Problem in the Design of Intelligent Machine. In Vanlehn, K. (ed.), *Architectures for Intelligence: The Twenty-Second Carnegie Mellon Symposium on Cognition*, Lawrence Erlbaum Associates, Hillsdale, 357-423.

Codd, E. F. (1970), A relational model of data for large shared data bank. *Communications of the ACM* 13 (6), 377-387.

Cohen, B. and Murphy, G. (1984), Models of Concepts. *Cognitive Science* 8, 27-58.

Collins, A. and Loftus, E. F. (1975), A spreading activation theory of semantic processing. *Psychological Review* 82, 407-428.

Collins, A. M. and Quillian, M. R. (1969), Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8, 240-247.

Conrad, C. (1972), Cognitive economy in semantic memory. *Journal of Experimental Psychology* 92, 149-154.

Dogac, A., YÜRÜTEN, B. and Spaccapietra, S. (1989), A Generalized Expert System for Database Design. *IEEE Transactions on Software Engineering* 15 (4), 479-491.

Domeshek, E. A., Herndon, M. F., Bennett, A. W. and Kolodner, J. L. (1994a), A Case-Based Design Aid for Conceptual Design of Aircraft Subsystems. In *Proceedings of the 10th Conference on AI for Application*, IEEE Computer Society Press, Washington, 63-69.

Domeshek, E. A., Kolodner, J. L. and Zimring, C. M. (1994b), The Design of a Tool Kit for Case-based Design Aids. In Gero, J. S. (ed.), *Proceedings of*

Artificial Intelligence in Design '94, Kluwer Academic Publishers, Netherlands.

Domeshek, E. and Kolodner, J. (1993), Using the Points of Large Cases. *AI EDAM* 7 (2), 87-96.

Dreyfus, H. L. and Dreyfus, S. E. (1986), *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. Free Press, New York.

Egan, D. E. and Schwartz, B. J. (1979), Chunking in recall of symbolic drawings. *Memory & Cognition* 7, 149-158.

Fairley, R. E. (1985), *Software Engineering Concepts*. McGraw-Hill, New York.

Gentner, D. (1989), Finding the needle: Accessing and reasoning from prior cases. In *Proceedings of a Workshop on Case-Based Reasoning*, Pensacola Beach, Florida 137-143.

Gick, M. L. and Holyoak, K. J. (1980), Analogical Problem Solving. *Cognitive Psychology* 12, 306-355.

Gick, M. L. and Holyoak, K. J. (1983), Schema Induction and Analogical Transfer. *Cognitive Psychology* 15 (1), 1-38.

Guindon, R. (1990), Knowledge exploited by experts during software system design. *Int. J. Man-Machine Studies* 33, 279-304.

Hammer, M. and McLeod, D. (1981), Database description with SDM: A semantic database model. *ACM transactions on Database Systems* 6 (3), 351-386.

Hammond, K. J. (1989), *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, San Diego.

Hampton, J. A. (1979), Polymorphous Concepts in Semantic Memory. *Journal of Verbal Learning and Verbal Behavior* 18, 441-461.

Harman, G. (1987), (Nonsolipsistic) Conceptual Role Semantics. In Lepore, E. (ed.), *New Directions in Semantics*. Academic Press, San Diego, 55-81.

Hennessy, D. H. and Hinckle, D. (1992), Applying Case-Based Reasoning to Autoclave Loading. *IEEE Expert* 7 (5), 21-26.

Hinrichs, T. R. (1992), *Problem Solving in Open Worlds: A Case Study in Design*. Lawrence Erlbaum Association, Hillsdale, New Jersey.

Hirschheim, R. (1985), Information systems epistemology: an historical perspective. In Mumford, E., Hirschheim, R. and Fitzberald, R. (eds.), *Research Methods in Information Systems*. North-Holland, Amsterdam, 13-38.

Hirschheim, R., Klein, H. K. and Lyytinen, K. (1995), *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*. Cambridge University Press, Cambridge.

Holyoak, K. J. and Koh, K. (1987), Surface and structural similarity in analogical transfer. *Memory and Cognition* 15 (4), 332-340.

Howard, R. W. (1987), *Concepts and Schemata: An Introduction*. Cassell, London.

Hull, R. and King, R. (1987), Semantic Database Modelling: Survey, Applications, and Research Issues. *ACM Computing Surveys* 19 (3), 201-260.

Johnson, P. E. (1983), What kind of expert should a system be? *Journal of Medicine and Philosophy* 8, 77-97.

Johnson, P. E., Duran, A. S., Hassebrock, F., Mollerand, J. and Prietula, M. (1981), Expertise and Error in Diagnostic Reasoning. *Cognitive Science* 5, 235-283.

Kintsch, W. and van Dijk, T. A. (1978), Toward a Model of Text Comprehension and Production. *Psychological Review* 85 (5), 363-394.

Klein, G. A. and Calderwood, R. (1988), How do people use analogues to make decision? In Kolodner, J. L. (ed.), *Proceedings of a Workshop on Case-Based Reasoning (DARPA)*, Clearwater, Florida, Morgan Kaufmann.

Klein, H. K. and Hirschheim, R. A. (1987), A Comparative Framework of Data Modelling Paradigms and Approaches. *The Computer Journal* 30 (1), 8-15.

Kolodner, J. L. (1983a), Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science* 7, 243-280.

Kolodner, J. L. (1983b), Reconstructive Memory: A Computer Model. *Cognitive Science* 7, 281-328

Kolodner, J. L. (1984), *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Kolodner, J. L. (1991), Improving Human Decision Making through Case-Based Decision Aiding. *AI Magazine* 12 (2), 52-68.

Kolodner, J. L. (1993), *Case-Based Reasoning*. Morgan Kaufmann, San Mateo.

Kolodner, J. L. (1996), Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning. In Leake, D. B. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. AAAI Press, Menlo Park, 349-370.

Kolodner, J. L. and Wills, L. M. (1993), Case-Based Creative Design. In *AAAI Spring Symposium on AI and Creativity*, Stanford, CA.

Leake, D. B. (1996), CBR in Context: The Present and Future. In Leake, D. B. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. AAAI Press, Menlo Park, 3-30.

Lesgold, A., Glaser, R., Rubinson, H., Klopfer, D., Feltovich, P. and Wang, Y. (1988), Expertise in a complex skill: Diagnosing X-ray pictures. In Chi, M. T. H., Glaser, R. and Farr, M. J.(eds.), *The Nature of Expertise*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 311-342.

Lewis, P. J. (1994), *Information-Systems Development: Systems Thinking in the Field of Information-systems*. Pitman Publishing, London.

Lloyd-Williams, M. and Beynon-Davies, P. (1992), Expert Systems for Database Design: A Comparative Review. *Artificial Intelligence Review* 6, 263-283.

Loucopoulos, P. and Zicari, R. (1992), *Conceptual Modeling, Databases, and CASE: An Integrated View of Information Systems Development*. John Wiley&Sons, New York.

Maher, M. L., Balachandran, M. B. and Zhang, D. M. (1995), *Case-Based Reasoning in Design*. Lawrence Erlbaum Association, Mahwah, New Jersey.

Malek, M. and Labbi, A. (1995), Integration of Case-Based Reasoning and Neural Networks Approaches for Classification. *TIMC-IMAG and LIFIA-IMAG*, Bat. lifia, Grenoble, France.

Mandler, J. M. (1984), *Stories, scripts and scenes: aspects of schema story*. Lawrence Erlbaum Association, Hillsdale, New Jersey.

Mandler, J. M. and Ritchey, G. H. (1977), Long-term memory for pictures. *Journal of Experimental Psychology: Human Learning and Memory* 3, 386-396.

Mandler, J. M. and Stein, N. L. (1974), Recall and recognition of pictures by children as a function of organization and distractor similarity. *Journal of Experimental Psychology* 102, 657-669.

Mannila, H. (1996), Schema Design and Knowledge Discovery. In *Keynote Address of 15th International Conference on Conceptual Modeling*, Cottbus Germany.

Mark, W., Simoudis, E. and Hinkle, D. (1996), Case-Based Reasoning: Expectations and Results. In Leake, D. B. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. AAAI Press, Menlo Park, 269-294.

Medin, D. L. and Schaffer, M. M. (1978), Context theory of classification learning. *Psychological Review* 85 (5), 207-238.

Medin, D. L. and Smith, E. E. (1984), Concepts and Concept Formation. *Annual Review of Psychology* 35, 113-138.

Mednick, S. A. (1962), The Associative Basis of the Creative Process. *Psychological Review* 69 (3), 220-232.

Mervis, C. B. (1980), Category Structure and the Development of Categorization. In Spiro, R. J., Bruce, B. C. and Brewer, W. F. (eds.), *Theoretical Issues in Reading Comprehension: perspectives from cognitive psychology, linguistics, artificial intelligence and education*. Lawrence Erlbaum Association, Hillsdale, New Jersey, 279-307.

Mervis, C. B. and Pani, J. R. (1980), Acquisition of Basic Categories. *Cognitive Psychology* 12, 496-522.

Mervis, C. B. and Rosch, E. (1981), Categorization of natural objects. *Annual Review of Psychology* 32, 89-115.

Millward, R. B. (1980), Models of concept formation. In Snow, R. E., Federico, P. and Montague, W. E. (eds.), *Aptitude, learning and instruction: cognitive process analyses of learning and problem solving*. Lawrence Erlbaum Association, Hillsdale, New Jersey.

Mandler, J. M. and Ritchey, G. H. (1977), Long-term memory for pictures. *Journal of Experimental Psychology: Human Learning and Memory* 3, 386-396.

Mandler, J. M. and Stein, N. L. (1974), Recall and recognition of pictures by children as a function of organization and distractor similarity. *Journal of Experimental Psychology* 102, 657-669.

Mannila, H. (1996), Schema Design and Knowledge Discovery. In *Keynote Address of 15th International Conference on Conceptual Modeling*, Cottbus Germany.

Mark, W., Simoudis, E. and Hinkle, D. (1996), Case-Based Reasoning: Expectations and Results. In Leake, D. B. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. AAAI Press, Menlo Park, 269-294.

Medin, D. L. and Schaffer, M. M. (1978), Context theory of classification learning. *Psychological Review* 85 (5), 207-238.

Medin, D. L. and Smith, E. E. (1984), Concepts and Concept Formation. *Annual Review of Psychology* 35, 113-138.

Mednick, S. A. (1962), The Associative Basis of the Creative Process. *Psychological Review* 69 (3), 220-232.

Mervis, C. B. (1980), Category Structure and the Development of Categorization. In Spiro, R. J., Bruce, B. C. and Brewer, W. F. (eds.), *Theoretical Issues in Reading Comprehension: perspectives from cognitive psychology, linguistics, artificial intelligence and education*. Lawrence Erlbaum Association, Hillsdale, New Jersey, 279-307.

Mervis, C. B. and Pani, J. R. (1980), Acquisition of Basic Categories. *Cognitive Psychology* 12, 496-522.

Mervis, C. B. and Rosch, E. (1981), Categorization of natural objects. *Annual Review of Psychology* 32, 89-115.

Millward, R. B. (1980), Models of concept formation. In Snow, R. E., Federico, P. and Montague, W. E. (eds.), *Aptitude, learning and instruction: cognitive process analyses of learning and problem solving*. Lawrence Erlbaum Association, Hillsdale, New Jersey.

Mingers, J. and Gill, A. (1997), *Multimethodology: The Theory and Practice of Combining Management Science Methodology*. John Wiley&Sons, Chichester.

Mizuno, Y. (1983), Software Quality Improvement. *IEEE Computer* 15 (3), 66-72.

Motta, E. (1997), Trends in knowledge modelling: report on the 7th KEML Workshop. *The Knowledge Engineering Review* 12 (2), 209-217.

Murphy, G. L. and Medin, D. L. (1985), The Role of Theories in Conceptual Coherence. *Psychological Review* 92 (3), 289-315.

Newell, A. (1982), The Knowledge Level. *Artificial Intelligence* 18, 87-127.

Norman, A. (1997), Why It's Good That Computers Don't Work Like the Brain. In Denning, P. J. and Metcalfe, R. M. (eds.), *Beyond Calculation: The next fifty years of computing*. Springer-Verlag, New York, 105-116.

Palmer, S. F. (1978), Fundamental aspects of cognitive representation. In Rosch, E. and Lloyd, B. B. (eds.), *Cognition and Categorization*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 259-303.

Patula, A. (1992), Case-Based Technology: The Path to Expertise. In *Proceedings of 1992 Information Resources Management Association International Conference*, Charleston, South Carolina, 97-102.

Peckham, J. and Maryanski, F. (1988), Semantic Data Model. *ACM Computing Surveys* 20 (3), 153-189.

Porter, B. W., Bareiss, R. and Holte, R. C. (1990), Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artificial Intelligence* 45, 229-263.

Quillian, R. (1968), Semantic memory. In Minsky, M. (ed.), *Semantic Information Processing*. MIT Press, MA, 217-270.

Reed, S. K. (1972), Pattern recognition and categorization. *Cognitive Psychology* 3, 382-407.

Reiner, D. (1992), Database Design Tools. In Batini, C., Ceri, S. and Navathe, S. B. (eds.), *Conceptual Database Design: An Entity-Relationship Approach*, The Benjamin/Cummings Publishing Company, Redwood, 411-451.

Rey, G. (1983), Concepts and stereotypes: A critical discussion of categories and concepts. *Cognition* 15, 237-262.

Riesbeck, C. K. (1996), What Next? The Future of Case-Based Reasoning in Post-Modern AI. In Leake, D. B. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, AAAI Press, Menlo Park, 371-388.

Rips, L. J. (1975), Inductive judgements about natural categories. *Journal of Verbal Learning and Verbal Behavior* 14, 665-681.

Rips, L. J., Shoben, E. J. and Smith, E. E. (1973), Semantic distance and the verification of semantic relations. *Journal of Verbal Learning and Verbal Behavior* 12, 1-20.

Rosch, E. (1975a), Cognitive representations of semantic categories. *Journal of Experimental Psychology: General* 104, 192-223.

Rosch, E. (1975b), Cognitive Reference Points. *Cognitive Psychology* 7, 532-547.

Rosch, E. and Mervis, C. B. (1975), Family resemblance studies in the internal structure of category. *Cognitive Psychology* 7, 573-605.

Rosch, E. H., Simpson, C. and Miller, R. S. (1976a), Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance* 2, 491-502.

Rosch, E.(1978), Principles of Categorization. In Rosch, E. and Lloyd, B. B. (eds.), *Cognition and Categorization*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 27-48.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M. and Boyes-Braem, P. (1976b), Basic Objects in Natural Categories. *Cognitive Psychology* 8, 382-439.

Ross, B. H. (1989), Some psychological results on case-based reasoning. In Hammod, K. (ed.), *Proceedings of a Workshop on Case-Based Reasoning (DARPA)*, Pensacola Beach, Florida, Morgan Kaufmann, 144-147.

Rumbaugh, J., Blaha, M., Permerlani, W., Eddy, F. and Lorensen, W. (1991), *Object-Oriented Modeling and Design*. Prentice-Hall, New York.

Rumelhart, D. E. (1980), Schemata: The Building Blocks of Cognition. In Spiro, R. J., Bruce, B. C. and Brewer, W. F. (eds.), *Theoretical Issues in Reading Comprehension: Perspectives from Cognitive Psychology, Linguistics, Artificial Intelligence, and Education*. Lawrence Erlbaum Association, Hillsdale, New Jersey, 33-58.

Schank, R. C. (1982), *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press, Cambridge.

Schank, R. C. and Abelson, R. (1977), *Scripts, plans, goals, and understanding.* Lawrence Erlbaum Association, Hillsdale, New Jersey.

Schreiber, A. (1992), *Pragmatics of the Knowledge Level.* University of Amsterdam, Department of Social Science Informatics. Ph.D thesis.

Schreiber, A., Akkermans, H. and Wielinga, B. (1990), On Problems with the Knowledge Level Perspective. In *Proceedings of the Banff-90 Knowledge Acquisitions for Knowledge-Based Systems Workshop*, University of Calgary, 30-1 - 30-14.

Schvaneveldt, R. W., Durso, F. T., Goldsmith, T. E., Breen, T. J. and Cooke, N. M. (1985), Measuring the structure of expertise. *Int. J. Man-Machine Studies* 23, 699-728.

Simpson, R. L. (1985), A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation. Technical Report GIT-ICS-85/18, Georgia Institute of Technology.

Skyrme, D. J. (1997), Multimethodology - the Knowledge Perspective. In Minger, J. and Gill, A. (eds.), *Multimethodology: The theory and Practice of Combining Management Science Methodology*. John Wiley&Sons, Chichester, 217-240.

Smith, E. E. and Medin, D. L. (1981), *Categories and Concepts*, Harvard University Press, London.

Srinivasan, A. and Te'eni, D. (1995), Modeling as Constrained Problem Solving: An Empirical Study of the Data Modeling Process. *Management Science* 41 (3), 419-434.

Storey, V. C. (1993), A selective survey of the use of artificial intelligence for database design systems. *Data&Knowledge Engineering* 11, 61-102.

Storey, V. C. and Goldstein, R. C. (1993), Knowledge-Based Approaches to Database Design. *MIS Quarterly*, 25-46.

Storey, V. C., Thompson, C. B. and Ram, S. (1995), Understanding database design expertise. *Data&Knowledge Engineering* 16, 97-124.

Storey, V. C., Yang, H. L. and Goldstein, R. C. (1996), Semantic integrity constraints in knowledge-based database design systems. *Data&Knowledge Engineering* 20, 1-37.

Tavolato, P. and Vincena, K. (1984), A Prototyping Methodology and Its Tools. In Budde, R. et al. (eds.), *Approaches to Prototyping*, Springer-Verlag, New York, 434-436.

Teorey, T. J., Yang, D. and Fry, J. P. (1986), A logical design methodology for relational databases using the Extended Entity-Relationship Model. *ACM Computing Surveys* 18 (2), 197-222.

Tulving, E. (1972), Episodic and semantic memory. In Tulving, E. and Donaldson, W. (eds.), *Organization of Memory*. Academic Press, New York, 382-403.

Tulving, E. (1985), How Many Memory Systems Are There? *American Psychologist*, 385-398.

Tversky, A. (1977), Features of Similarity. *Psychological Review* 84 (4), 327-352.

Tversky, B. and Hemenway, H. (1983), Categories of environmental scenes. *Cognitive Psychology* 15, 121-149.

Williams, M. D. and Hollan, J. D. (1981), The Process of Retrieval from Very Long-Term Memory. *Cognitive Science* 5, 87-119.

Winograd, T. and Flores, F. (1987), *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley Publishing Company, Menlo Park.

Wittgenstein, L. (1958), *Philosophical Investigations*. Basil Blackwell, Oxford.

Appendix A

The Meta-knowledge of the Key Recalling Process

This appendix presents the meta knowledge of the key recalling process, which is concerned with how to locate knowledge in the DBKM.

A.1 The Meta-knowledge of the First Part of the Key Recalling Process: the first-pass focus sub-process

Assume that can see all the domain level cases?

Initiating Rules: The first type of control knowledge is for the first-pass focus sub-process and consists of four general rules, being performed by the compulsory sequence of the temporal activities as shown in Figure 5.4.

G-Rule 1.1: If a specific UoD is initiated,

then the typical case of the specified UoD is activated.

This rule is carried out after initiating a specific UoD by means of the typical exemplar link of that initiated UoD in order to enable a first good shot at the appropriate case.

G-Rule 1.2 : If the typical case of the specified UoD is activated,
then the relevant concepts in the concept base that are of the correspondences of the entities in the activated case are activated.

This mapping is guaranteed by the relevance of the concept and case bases discussed in section 4.2.3.

G-Rule 1.3 : If the typical case of the specified UoD and
the corresponding concepts of the activated entities in the concept base are activated,
then the relevant relations in the concept base that are of the correspondences of the relationships in the activated case are activated.

This is for the purpose of eliminating/hiding the irrelevant relations of the activated concepts by referring the relationships between/among entities in the typical case. These two types of activated knowledge serve as reference points to retrieve other relevant cases.

G-Rule 1.4 : If the typical case of the specified UoD in the case base is activated and the corresponding part of the concept base is activated,

then the names of entities in the activated case (the corresponding concepts of the activated part of the concept base) are displayed.

Together, these are used for entry points in order to form the appropriate contexts to guide and focus the direction of the recalling.

A.2 The Meta-knowledge of the Second Part of the Key Recalling Process: the recalling cycle

The second type of control knowledge is for the recalling cycle and is embedded into the respective sub-processes discussed in section 5.2.1.2 as shown in Figure 5.5.

A.2.1 Eliciting Rules

Depending upon the types of user requirements, two kinds of general eliciting rules are available.

G-Rule 2.1 : If the new-added-entity type and the relevant displayed entity are clicked,

then the related concepts of the clicked entity (concept) are activated, indicating that their exemplars (entities) are not in the working case.

G-Rule 2.2 : If the deleted-entity type and the relevant displayed entity are clicked,
then the related concepts of the clicked entity (concept) are displayed, indicating that their exemplars (entities) are in the working case, or/and
the hidden relation(s), if any, of the non-deleted concepts are activated.

The above rules are for the *second-pass focus sub-process*. The mapping from the working case to the concept base is guaranteed by the *subjective* and *bijective* functions from the case base to the concept base, as discussed in section 4.2.3.

A.2.2 Indexing Rules

According to the situations which occur in the previous sub-process, two kinds of general indexing rules will be used to form the index set.

G-Rule 3.1 : If the new-added-entity type and the relevant displayed concept(s) is (are) confirmed,

then the confirmed concept(s) are packed into an indexing set as follows:

$$\left\{c_{n-ci}, r(c_{n-ci}, c_{r-dj}) \mid i, j \in N\right\}$$

c_{n-ci} : New - Confirmed Concept Name

c_{r-dj} : Related - Displayed Concept Name

r(.,.) : r1 or r2 Relation.

G-Rule 3.2 : If the deleted-entity type and the relevant displayed concept(s) or the activated, confirmed relation(s) of the non-deleted concepts are confirmed,

then the confirmed, deleted concept(s) and non-deleted concepts (relations) are packed into an indexing set including three parts as follows:

The first part is for the *deleted concepts*:

$$\left\{c_{di}, r(c_{di}, c_{r-dj}) \mid i, j \in N\right\}$$

c_{di} : Deleted Concept Name

c_{r-dj} : Related - Displayed Concept Name

r(.,.) : r1 or r2 Relation

The second part is for the *non-deleted concepts*:

$$\left\{c_{n-di}, r(c_{n-di}, c_{n-dj}) \mid i, j \in N\right\}$$

c_{n-di} : Non – Deleted Displayed Concept Name

r(.,.) : r1 or r2 Relation

The third part is for the *newly activated, confirmed relation(s)* of the non-deleted concepts

$$\left\{r(c_{n-di}, c_{n-dj}) | i, j \in N\right\}$$

c_{n-di} : Non – Deleted Dispalyed Concept Name

r(.,.) : r1 or r2 Relation

In the *transfer sub-process*, the confirmed concept(s) will first be packed into the appropriate context and will then be transferred into the case base to guide the direction of the retrieval of relevant case(s).

A.2.3 Finding Rules

Two main inherent properties of the relations (*sub-super* and *overlp* relations) in the case base are used to guide and stop the finding. These are the *bi-direction* of the links (subset, supset and overlp link) and the *strict partial order*. Because of the property of bi-direction of the links, the direction of search will be activated according to the types of user requirements. Because the *proper subset-superset relation* is a *strict partial order*, the notions of supremum (the least upper bound, abbreviated as sup) and infimum (the greatest lower bound, abbreviated as inf), based on the finding element, can be used as criteria for stopping the *find sub-process*.

A.2.3.1 The finding-rules of the new-added-entity type of user requirements

Before the find sub-process, the incoming index pack will be partitioned into some appropriate context, called the finding elements, if the index pack includes more than one object or relation. The set includes the appropriate context(s) called the finding set. For example, if the index pack is {d,f,r1(d,a),r1(f,c)}, the finding elements are {d,r1(d,a)}, {f,r1(f,c)} and {d,f,r1(d,a),r1(f,c)} and the finding set is {{d,r1(d,a)}, {f,r1(f,c)}, {d,f,r1(d,a),r1(f,c)}}. These finding elements are used to guide the direction of the retrieval of relevant cases.

G-Rule 4.1.1 : If a direct superset-case of the working case includes the maximum finding element in the finding set,
then the direct super-case including the maximum finding element of the working case is marked.

An example of the general rule is depicted in Figure A.1.

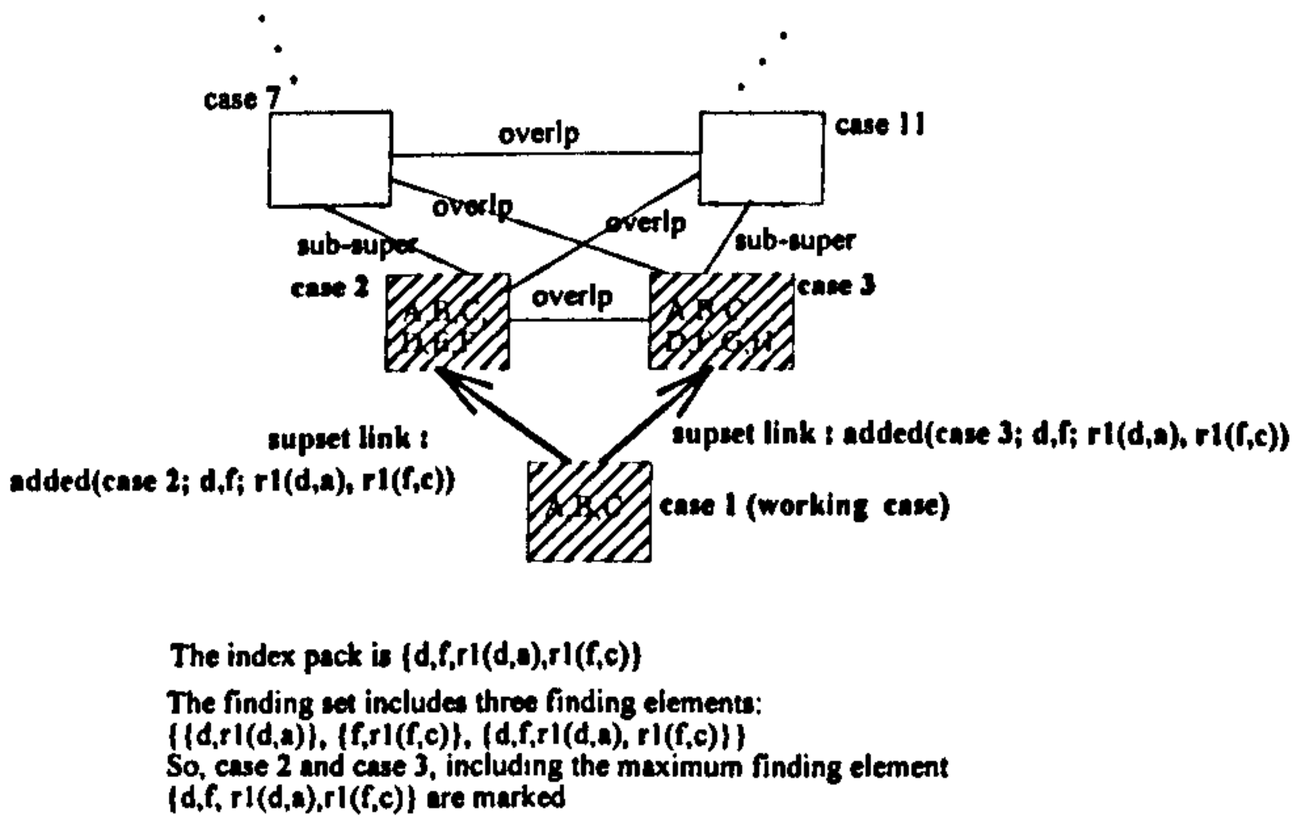


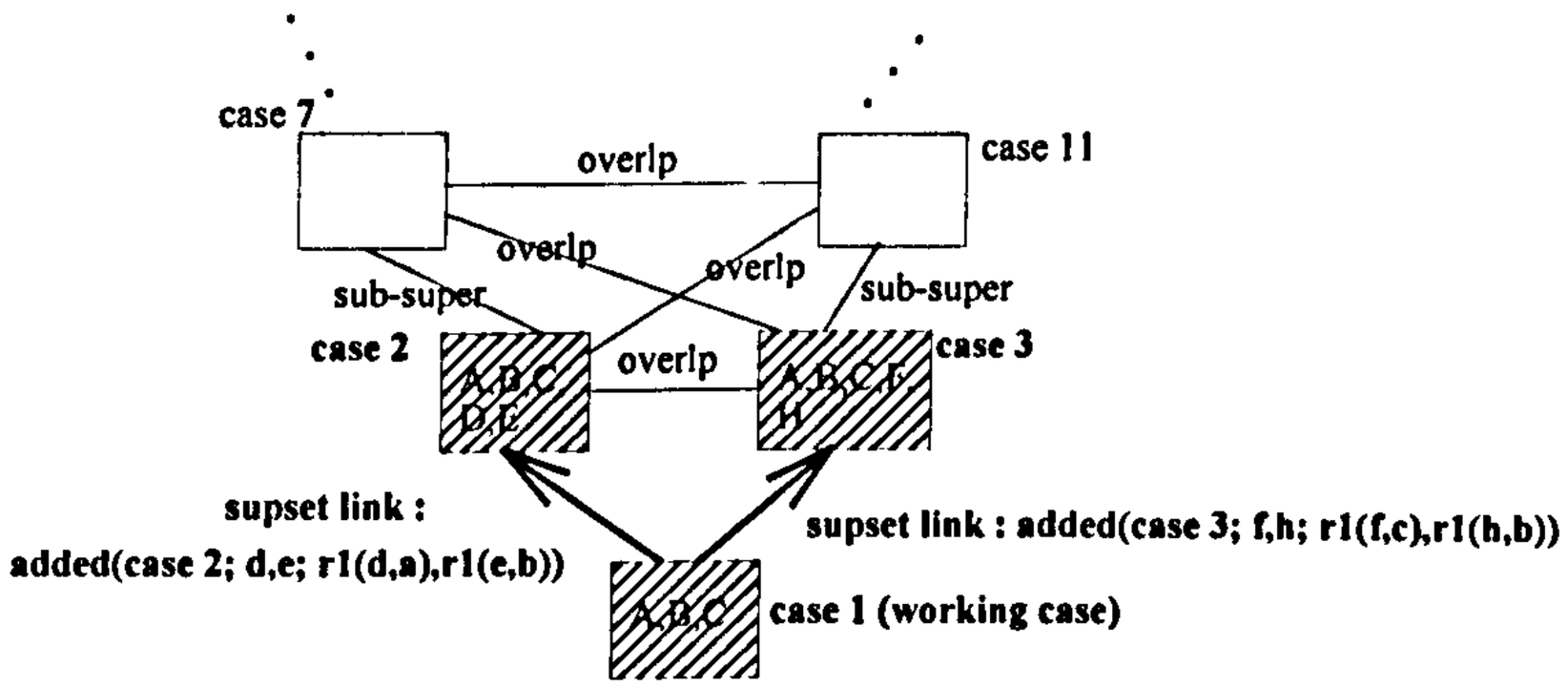
Figure A.1 : An example of G-Rule 4.1.1

By means of the supset links of the working case, the direct supset-cases, including the maximum finding element of the working case, will be marked.

G-Rule 4.1.2 : If the direct superset cases of the working case include the finding elements and cover the content of the maximum finding element, but do not include the maximum finding element,

then the direct superset cases, including the finding elements of the working case, are marked.

An example of the general rule is depicted in Figure A.2.



1. The original index pack is {d,f, r1(d,a), r1(f,c)}.
   So, the finding set includes three finding elements: {{d, r1(d,a)}, {f, r1(f,c)},{d, f, r1(d,a), r1(f,c)}}
2. The finding element {d, r1(d,a)}, is found out in case 2 and the finding element {f, r1(f,c)}, is found out in case 3
3. Case 2 and case 3 are marked.

Figure A.2 : An example of G-Rule 4.1.2

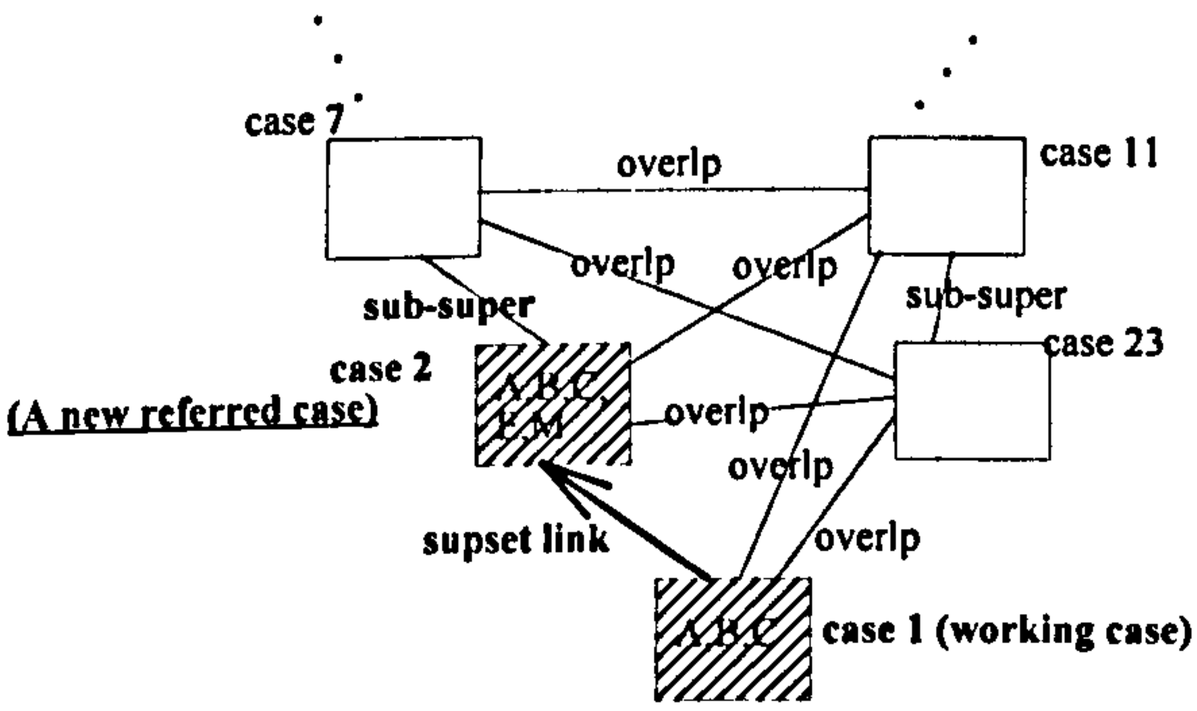
While the finding elements are spread over the direct supset-cases of the working case and no direct supset-case of the working case includes the maximum finding elements, the above rule will be used.

G-Rule 4.1.3 : If the content(s) of the direct supset-case(s) of the working case does(do) not include any finding elements in the finding set,

then a direct supset-case of the working case is activated as a new referred case (i.e. a new reference point) to find the relevant case(s) including the finding element(s).

There are two situations which occur as follows:

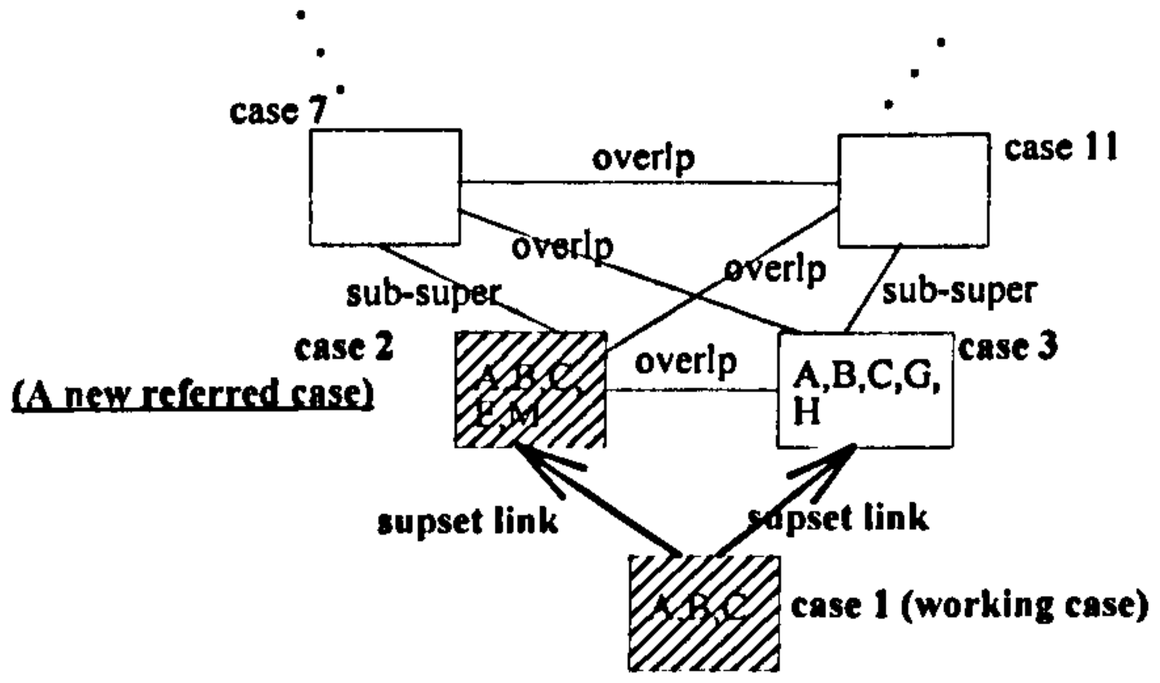
Situation One: There is only one direct supset-case of the working case. An example of the general rule is depicted in Figure A.3.



1. The original index pack is {d,f,r1(d,a),r1(f,c)}
   So the finding set includes three finding elements:
   {{d,r1(d,a)}, {f,r1(f,c)},{d,f,r1(d,a),r1(f,c)}}.
2. The direct supset-case of case 1- case 2 - does not include the finding elements.
3. Case 2 is activated as a new referred case (i.e. a new reference point) to find the relevant cases including the finding elements.

Figure A.3 : An example of G-Rule 4.1.3 (situation one)

Situation Two: There is more than one direct supset-case of the working case. An example of the general rule is depicted in Figure A.4.



1. The original index pack is {d,f, r1(d,a), r1(f,c)}
   So the finding set includes three finding elements:
   {{d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}}.

2. There are not any direct supset-cases of case 1 including the finding elements.

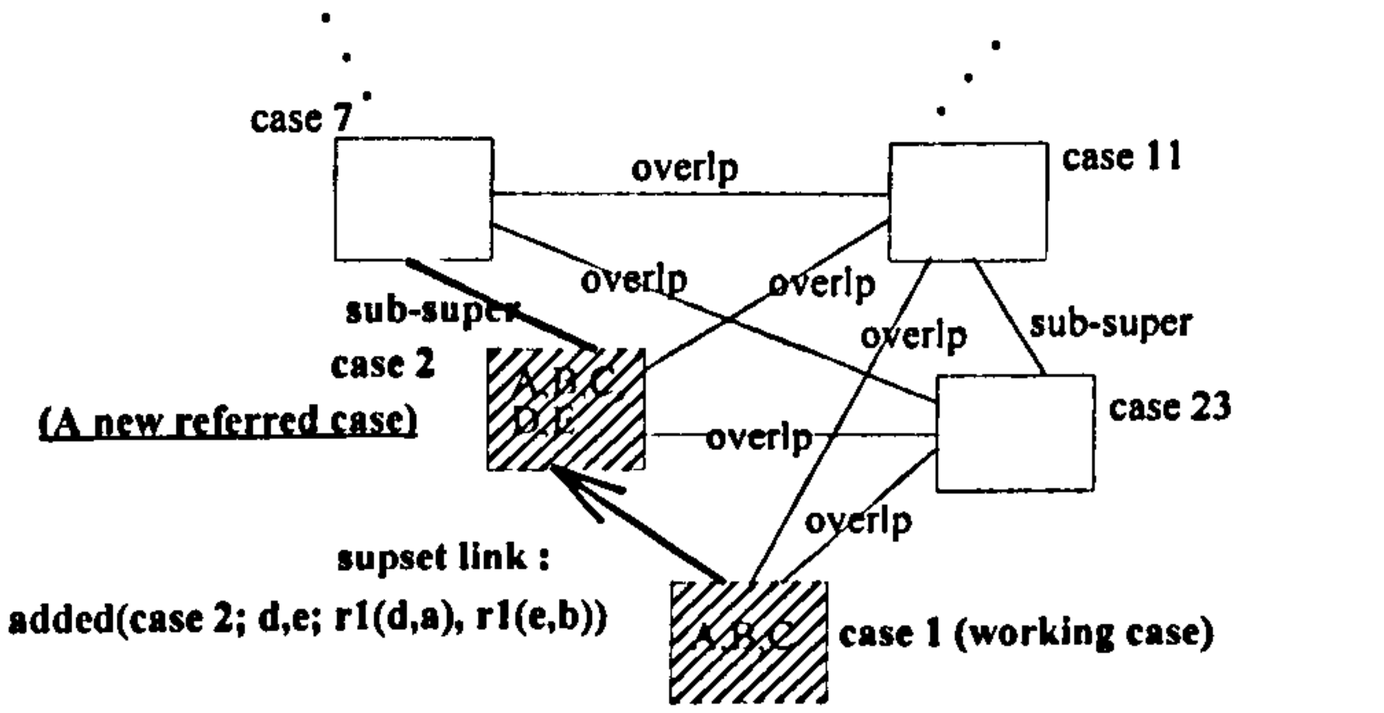
3. Case 2 is activated as a new referred case (i.e. a new reference point) to find the relevant cases including the finding elements.

Figure A.4 : An example of G-Rule 4.1.3 (situation two)

G-Rule 4.1.4 : If the content(s) of the direct supset-case(s) of the working case includes(include) some finding elements but do not cover the content of the maximum finding element in the finding set,

then a direct supset-case of the working case is activated as a new referred case (i.e. a new reference point) to find the relevant case(s) including the finding element(s). There are two situations which occur as follows:

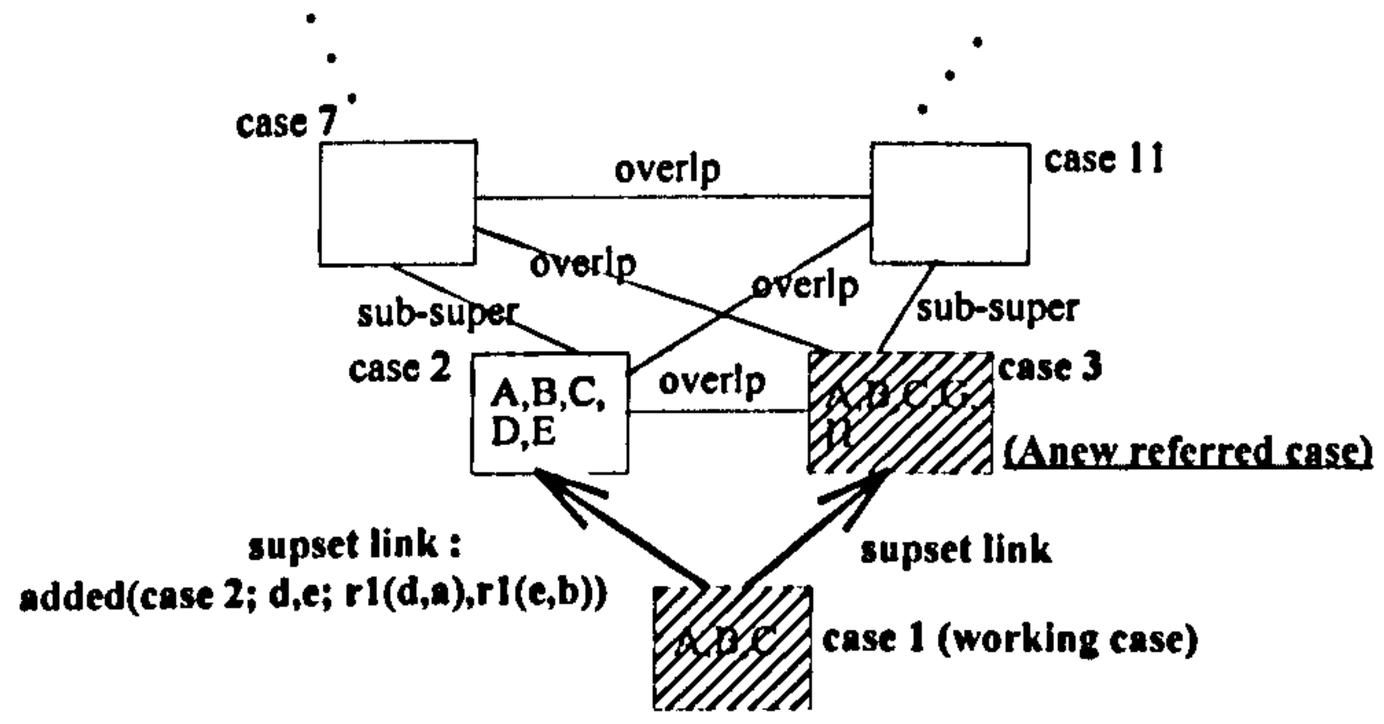
Situation One: There is only one direct supset-case of the working case. An example of the general rule is depicted in Figure A.5.



1. The original index pack is {d,f, r1(d,a), r1(f,c)}.
   So the finding set includes three finding elements:
   {{d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}}.
2. The direct supset-case of case 1- case 2 - includes the finding element {d, r1(d,a)} but not the maximum finding element.
3. There are missing elements: {f, r1(f,c)}, {d,f, r1(d,a), r1(f,c)}.
4. Case 2 is activated as a new referred case (i.e. a new reference point) to find the relevant cases including the missing elements.

Figure A.5 : An example of G-Rule 4.1.4 (situation one)

Situation Two: There is more than one direct supset-case of the working case. An example of the general rule is depicted in Figure A.6.



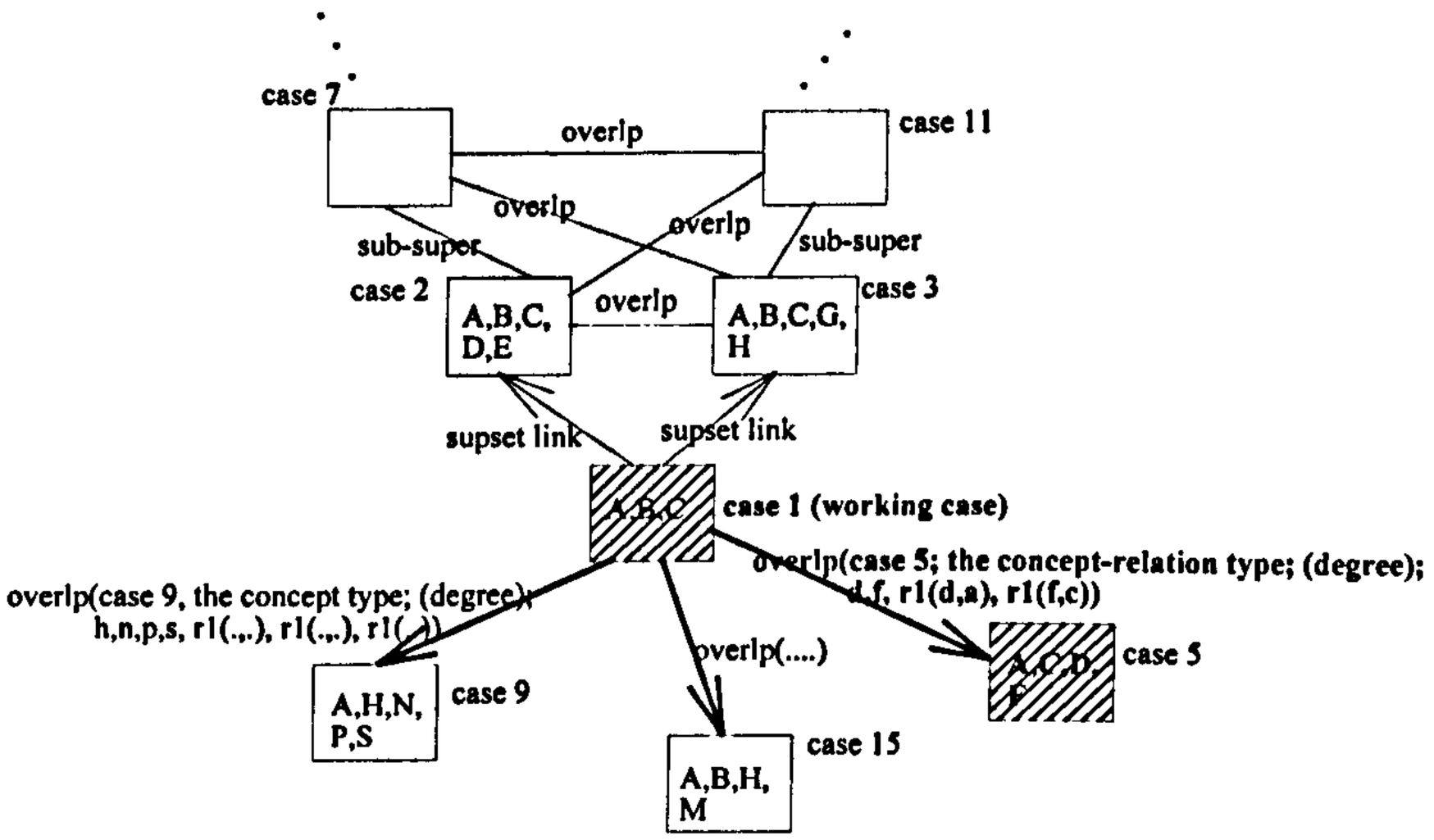
1. The original index pack is {d,f, r1(d,a), r1(f,c)}.
   So the finding set includes three finding elements:
   {{d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}}.
2. There are two direct supset-cases of case 1. Case 2 includes the finding element {d, r1(d,a)} and case 3 does not include any finding elements in the finding set.
3. There are missing elements {f, r1(d,a)}, {d,f, r1(d,a), r1(f,c)}.
4. Case 3 is activated as a new referred case (i.e. a new reference point) to find the relevant cases including the missing elements.

Figure A.6: An example of G-Rule 4.1.4 (situation two)

G-Rule 4.1.5 : If an overlapped case of the working case includes the maximum finding element in the finding set,

then the overlapped case(s), including the maximum finding element, of the working case is (are) marked.

An example of the general rule is depicted in Figure A.7.



1. The finding set includes three finding elements:
   {{d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}}.
2. So case 5, including the maximum finding element, is marked.

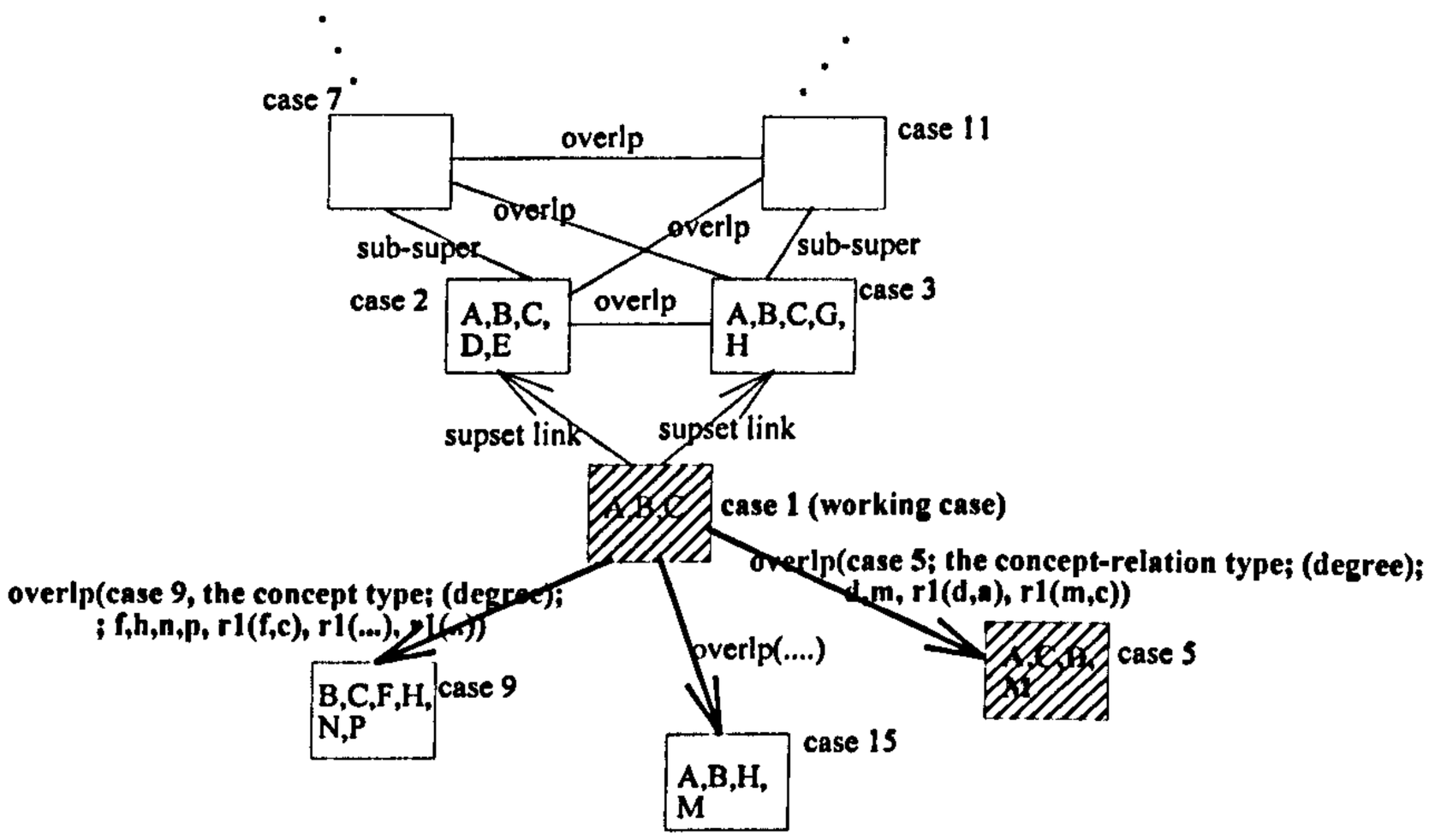
Figure A.7 : An example rule G-Rule 4.1.5

By means of the overlapped links of the working case, the overlapped cases, including the maximum finding element of the working case, will be marked.

G-Rule 4.1.6 : If the overlapped cases of the working case include some finding elements and cover the content of the maximum finding element, but do not include the maximum finding element,

then the overlapped cases, covering the maximum finding elements, of the working case are marked.

An example of the general rule is depicted in Figure A.8.



1. The finding set includes three finding elements:
   {{d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}}.
2. The finding element {d, r1(d,a)} is in case 9 and the finding element {f, r1(f,c)} is in case 5.
3. Case 5 and case 9 are marked.

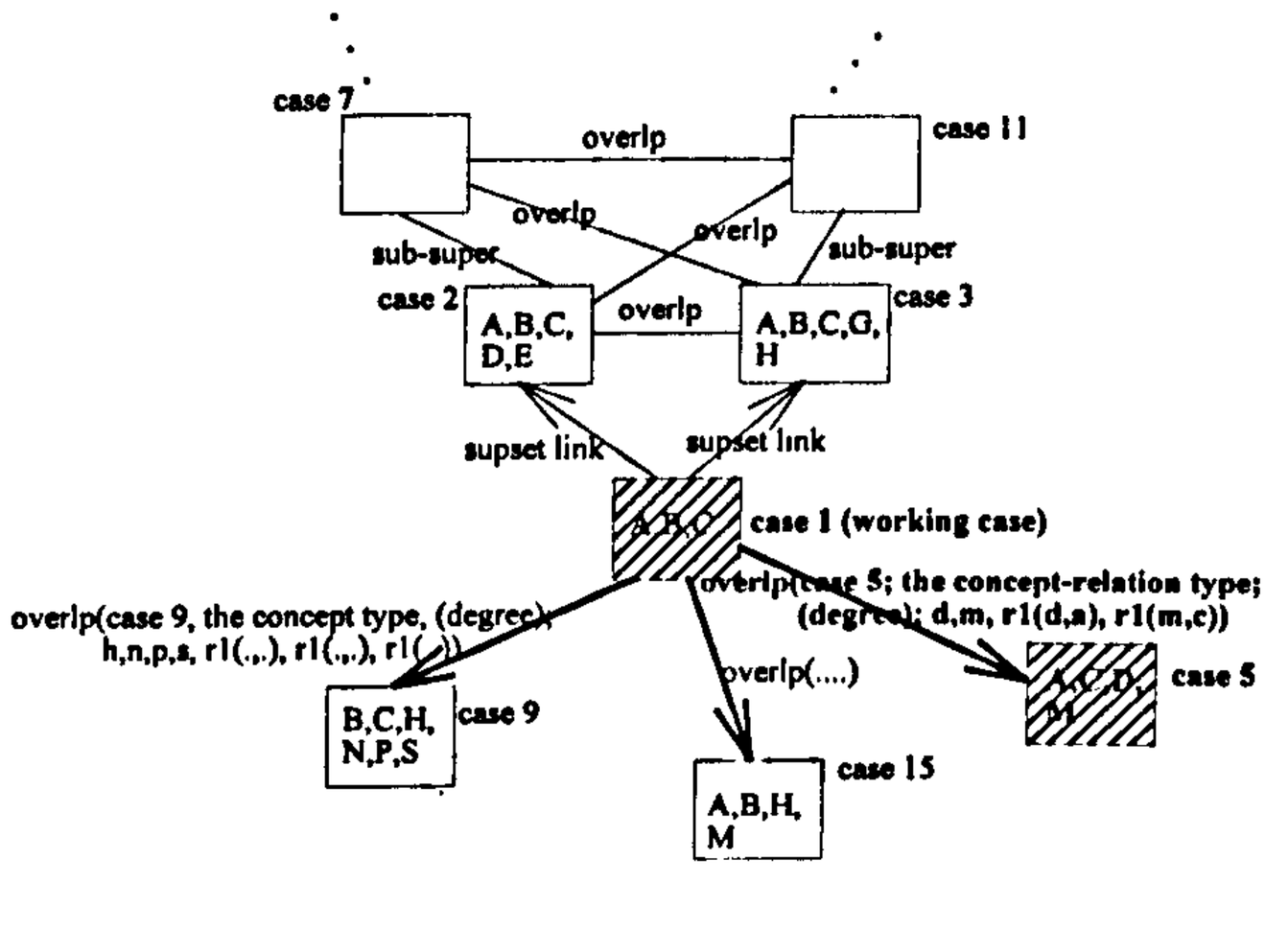
Figure A.8 : An example of G-Rule 4.1.6

While the finding elements are spread over the overlapped cases of the working case and no overlapped case of the working case includes the maximum finding element, the above rule will be used.

G-Rule 4.1.7 : If the content(s) of the overlap links of the working case just partially include/includes some finding elements but does/do not cover the content of the maximum finding element,

then the overlapped case(s), including some finding elements, of the working case is (are) marked.

An example of the general rule is depicted in Figure A.9.



1. The finding set includes three finding elements:
   {{d, rl(d,a)}, {f, rl(f,c)},{d,f, rl(d,a), rl(f,c)}}.
2. Only the finding element {d, rl(d,a)} is found in case 5 and the content of case 5 does not cover the maximum finding element
3. Case 5 is marked.

Figure A.9 : An example of G-Rule 4.1.7

A.2.3.2 The finding-rules of the deleted-entity type of user requirements

Before the find sub-process, the first part of the incoming index pack (i.e. the deleting concepts[67]) will be partitioned into some appropriate context called the *deleting elements* if the first part of the index pack includes more than one object or relation. The third part of the incoming index pack (i.e. the newly confirmed relation(s) of the non-deleted concepts), if it exists, will be partitioned into some

[67] The number of deleting concepts must be controlled by some proportion of the number of concepts in the typical case, for example one third of the number of concepts, otherwise the 'typical case' is not typical, but atypical.

appropriate context called the *finding elements* if this part includes more than one newly confirmed relation. The deleting element(s) with the non-deleted concepts (relations) together form the appropriate context called *the deleting-finding set.* For example, if the index pack is {{d,f,r1(d,a),r1(f,c)}; {a,b,c,r1(a,c),r1(a,b)};Φ}, the *deleting elements* are {d,r1(d,a)}, {f,r1(f,c)} and {d,f,r1(d,a),r1(f,c)} and the *deleting-finding set* is {{d,r1(d,a)},{f,r1(f,c)},{d,f,r1(d,a),r1(f,c)}; {a,b,c,r1(a,c),r1(a,b)}}. The elements in this set are used to guide the direction of the retrieval of relevant cases by means of the subset link(s) of the working case. By retrieving the relevant cases, the second part of the set (i.e. the non-deleting concepts and relations{a,b,c,r1(a,c),r1(a,b)}) is used as a precondition during the whole sub-process, and then the first part of the case (i.e. the deleting elements {d,r1(d,a)},{f,r1(f,c)},{d,f,r1(d,a),r1(f,c)}) is used to find the relevant cases. The finding element(s) with the non-deleted concepts (relations) together form the appropriate context called the finding set[68]. The elements in the finding set are used to guide the direction of the retrieval of relevant cases by means of the overlapped link(s) of the working case.

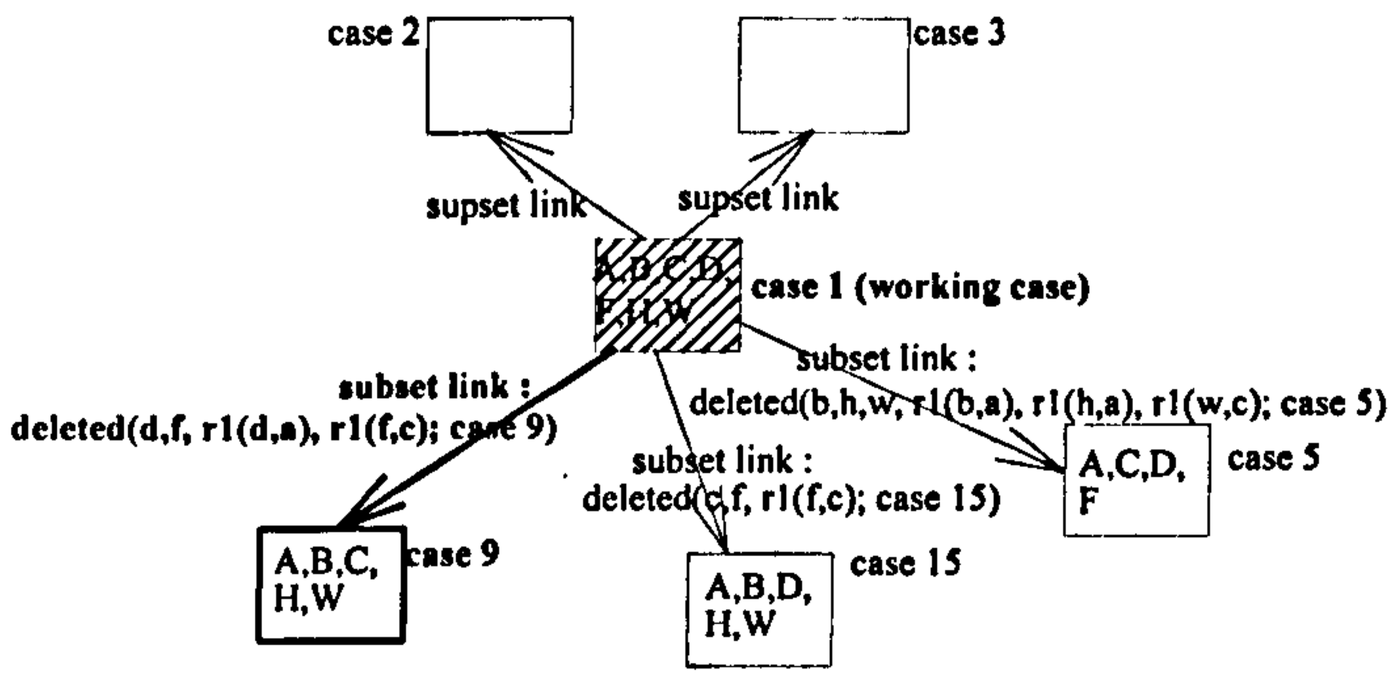
G-Rule 4.2.1 : If a direct subset-case of the working case does not include the maximum deleting element, but includes non-deleted concepts(relations),

then the direct subset-case is marked.

---

[68] This situation is more complicated than the forming of the deleting-finding set. The example will be given in rules 4.2.4 and 4.2.5 and will be discussed later.

An example of the general rule is depicted in Figure A.10.



1. The incoming index pack is
   {{d,f,r1(d,a),r1(f,c)}; {a,b,c,h,w,r1(a,b),r1(a,h),r1(a,c),r1(c,w)}}.
2. The deleting elements are
   {d,r1(d,a)}, {f,r1(f,c)} and {d,f,r1(d,a),r1(f,c)}.
3. The deleting-finding set is
   {{d,r1(d,a)},{f,r1(f,c)},{d,f,r1(d,a),r1(f,c)};
   {a,b,c,h,w,r1(a,b),r1(a,h),r1(a,c),r1(c,w)}}.
4. Case 9 does not include the maximum deleting element
   {d,f,r1(d,a),r1(f,c)}, but includes the second part of the deleting-finding set (non-deleting concepts and relations : {a,,b,c,h,w,r1(a,b),r1(a,h), r1(a,c),r1(c,w)}.
5. So case 9 is marked.

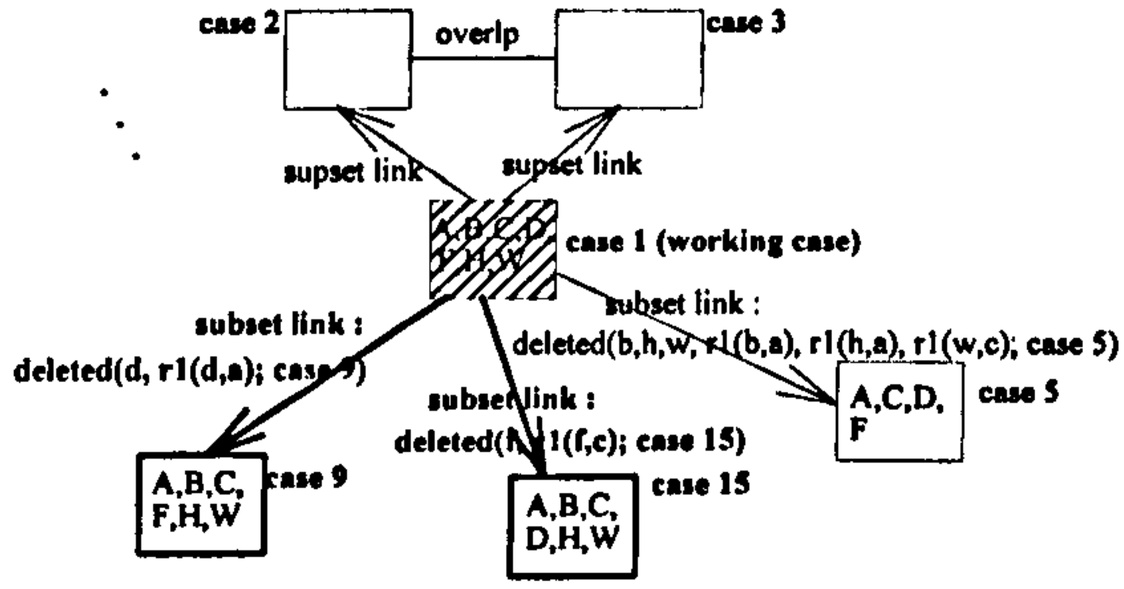
Figure A.10 : An example of G-Rule 4.2.1

By means of the subset links of the working case, the direct subset-case, which does not include the maximum deleting element, but includes the second part of the deleting-finding set (i.e. the non-deleting concepts and relations), is marked.

G-Rule 4.2.2 : If the direct subset-cases, which include the non-deleted concepts (relations) of the working case, do not include some deleting elements, but none of these direct subset-cases totally satisfies the deleted requirement (i.e. the contents of these cases include some deleting elements),

then the direct subset-cases, which partially satisfy the requirements, are marked.

An example of the general rule is depicted in Figure A.11.



1. The incoming index pack is
   {{d,f, r1(d,a), r1(f,c)}; {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}}.
2. The deleting elements are
   {d, r1(d,a)}, {f, r1(f,c)} and {d,f, r1(d,a), r1(f,c)}.
3. The deleting-finding set is
   {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}}.
   {{d, r1(d,a)},{f, r1(f,c)},{d,f, r1(d,a), r1(f,c)},
4. Case 9 and case 15 do not include the maximum deleting element {d,f, r1(d,a), r1(f,c)}, but the contents of these two cases cover the maximum deleting finding and include the second part of the deleting-finding set: {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}.
5. So case 9 and case 15 are marked

Figure A.11 : An example of G-Rule 4.2.2

While the deleting elements are spread over the direct subset-cases, including the second part of the deleting-finding set (i.e. the non-deleting concepts and relations) of the working case, and while none of these direct subset-cases totally satisfies the deleted requirement, the above rule will be used.

G-Rule 4.2.3 : If the direct subset-case(s), including the second part of the deleting-finding set of the working case, does/do include some deleting elements but none of it (them)

totally satisfies (satisfy) the deleted requirement (i.e. the content(s) of it (them) includes/include some deleting elements),

then a direct subset-case, including the non-deleting concepts (relations), of the working case is activated as a new referred case (i.e. a new referred point) to find the relevant case(s) that does/do not include the maximum deleting element. There are two situations which occur as follows:

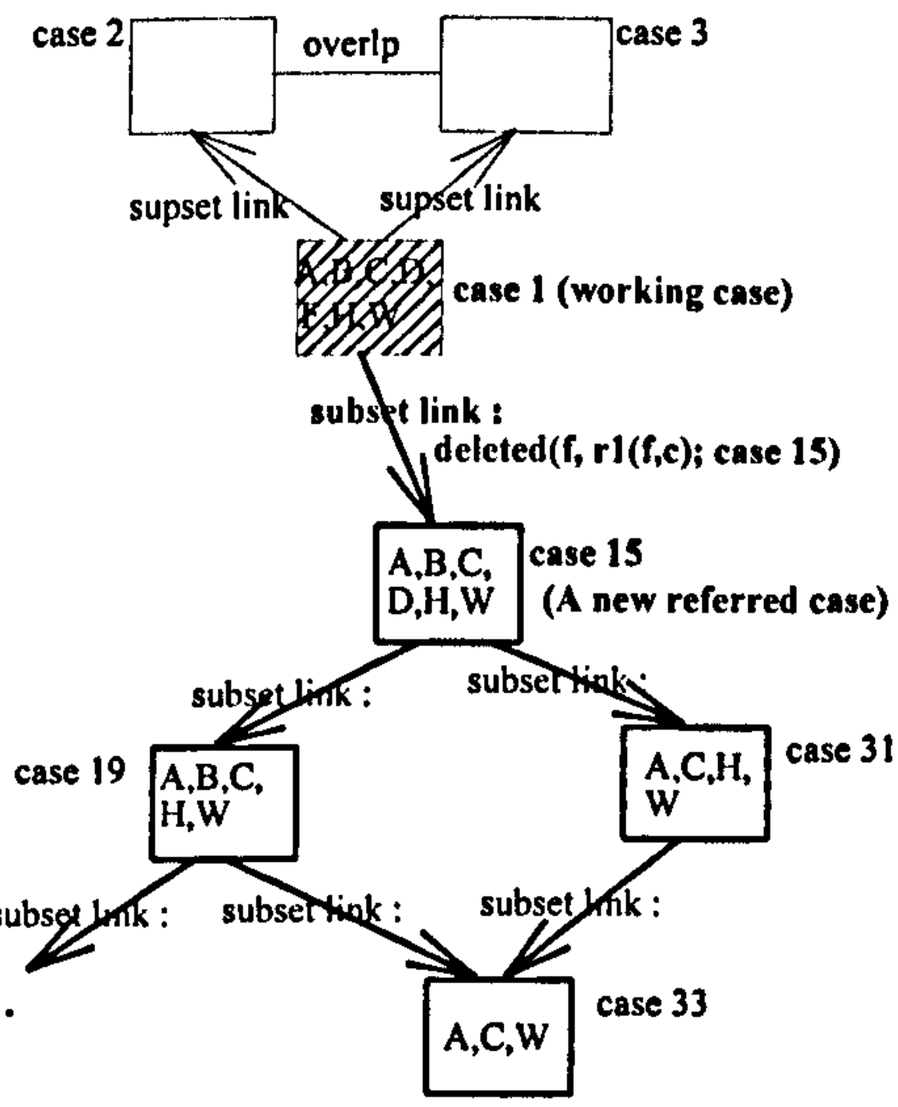
Situation One: There is only one direct subset-case of the working case (i.e. this direct subset-case satisfies the second part of the deleting-finding set and includes some deleting element, but not the maximum deleting element in the first part of the deleting-finding set). An example of the general rule is depicted in Figure A.12.

Situation Two: There is more than one direct subset-case of the working case (i.e. these direct subset-cases all satisfy the later part of the deleting-finding set and include some deleting elements[69]). The criterion of activating a direct subset-case of the working case is decided by the number of deleting concepts that these direct subset-cases include. A direct subset-case which includes the

---

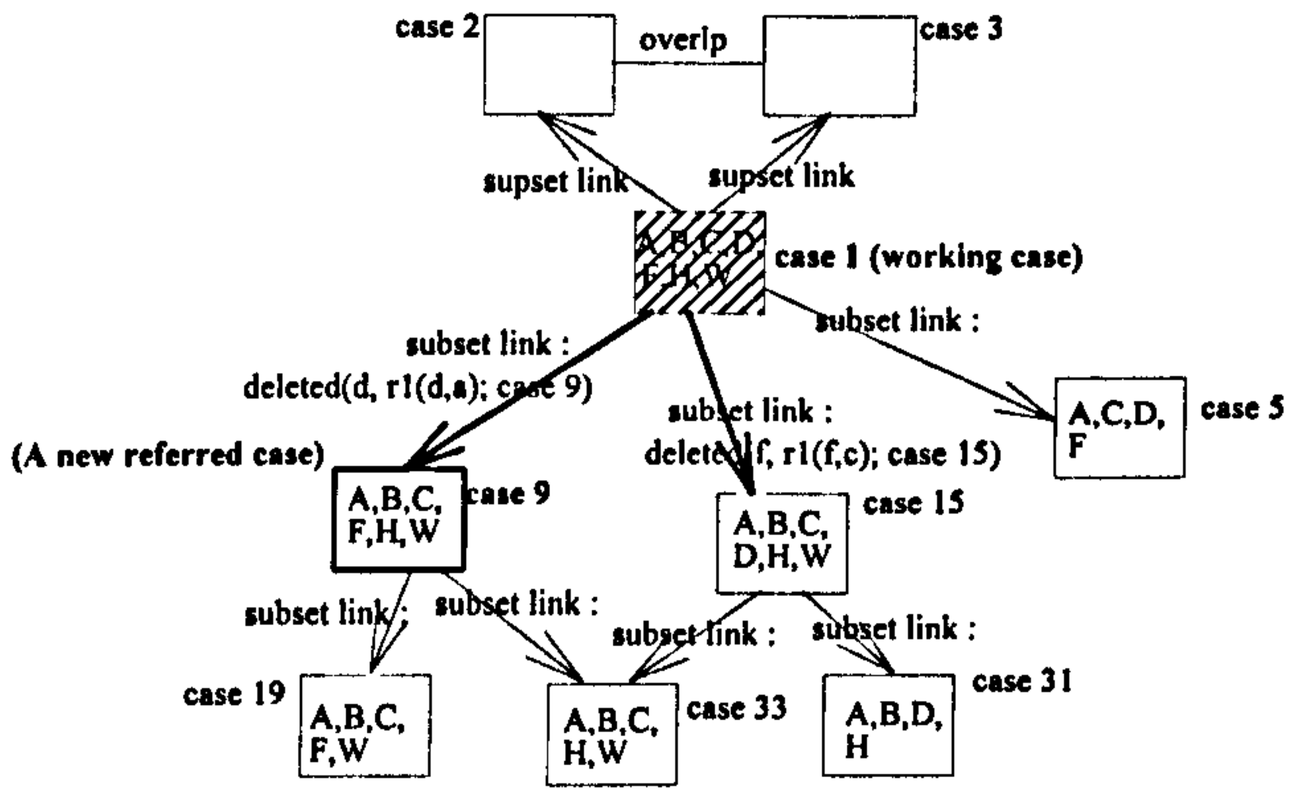
[69] This situation is applied to whether the contents of the direct subset-cases together cover the maximum deleting element or not. This means the condition of rule 4.2.2 also fits this situation.

minimum number of deleting concepts will be activated as a new referred case. If all these direct subset-cases have the same number of deleting concepts, one of the direct subset-cases will be activated randomly. For example, in Figure A.13, case 9 includes the deleting concept F, and case 15 includes the deleting concept D. They have the same number of deleting concepts. Thus, case 9 is selected randomly as a new referred point to find the relevant cases which totally satisfy the deleted requirement.



1. The incoming index pack is
   {{d,f, r1(d,a), r1(f,c)}; {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}}.
2. The deleting elements are
   {d, r1(d,a)}, {f, r1(f,c)} and {d,f, r1(d,a), r1(f,c)}.
3. The deleting-finding set is
   {{d, r1(d,a)},{f, r1(f,c)},{d,f, r1(d,a), r1(f,c)};
   {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}}.
4. Case 15 does not include the maximum deleting element
   {d,f, r1(d,a), r1(f,c)}, but includes the second part of the deleting-finding set (non-deleting concepts and relations: {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c),r1(c,w)}.
5. There is a missing deleting element {f, r1(f,c)}.
   point) to find the relevant case that does not include the missing
6. So case 15 is activated as a new referred case (i.e. a new reference element.

Figure A.12 : An example of G-Rule 4.2.3 (situation one)



1. The incoming index pack is
   {{d,f, r1(d,a), r1(f,c)}; {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(w,c)}}.
2. The deleting elements are
   {d, r1(d,a)}, {f, r1(f,c)},{d,f, r1(d,a), r1(f,c)}.
3. The deleting-finding set is
   {{d, r1(d,a)}, {f, r1(f,c)}, {d,f, r1(d,a), r1(f,c)}};
   {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(w,c)}}.
4. Case 9 and case 15 do not include the maximum finding element
   {d,f, r1(d,a), r1(f,c)}, but include the second part of the deleting-finding set: the non-deleting concepts and relations,
   {a,b,c,h,w, r1(a,b), r1(a,h), r1(a,c), r1(c,w)}.
5. Case 9 includes the deleting concept F, and case 15 includes the deleting concept D. They have got the same number of deleting concepts.
6. One of these direct subset-cases of the working case, for example, case 9, is activated randomly as a new referred case (i.e. a new referred point) to find the case which totally satisfies the deleted requirement.

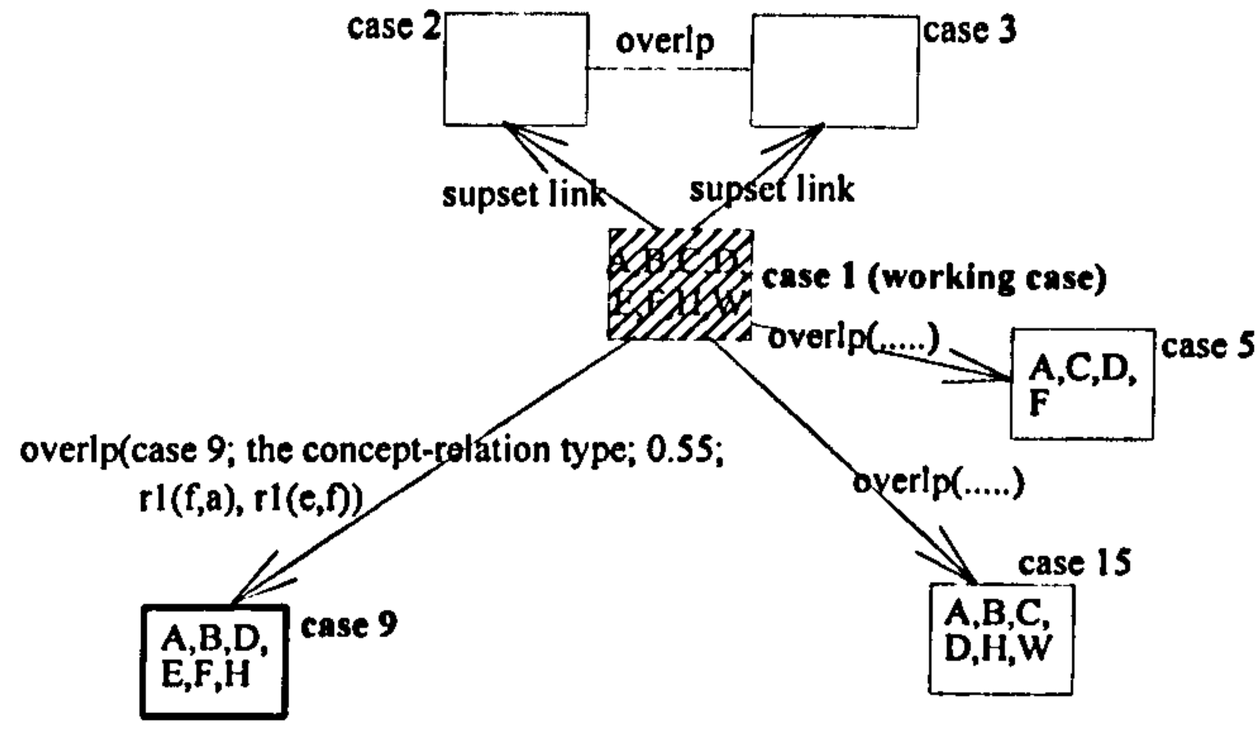
Figure A.13 : An example of G-Rule 4.2.3 (situation two)

The following three rules (G-Rules 4.2.4, 4.2.5, and 4.2.6) are used while the third part of the incoming index pack is not an empty set (i.e. there exist newly activated confirmed relations(s) of the non-deleted concepts).

G-Rule 4.2.4 If an overlapped case of the working case includes the maximum finding element and the non-deleted concepts (relations) in the finding set,

then the overlapped case(s) of the working case, which includes (include) the maximum finding element, is (are) marked.

An example of the general rule is depicted in Figure A.14.



1. The incoming index pack is {{c,w, r1(c,a), r1(w,c), r1(c,e)};
   {a,b,d,e,f,h, r1(a,b), r1(a,d), r1(a,h)}; {r1(f,a), r1(e,f)}}
2. The finding elements are {r1(f,a)} and {r1(e,f), r1(f,a)}
3. Case 9 includes the maximum finding element {R(E,F),R(F,A)}
4. So case 9 is marked.

Figure A.14 : An example of G-Rule 4.2.4

The general view of the main body of the working case (case 1), represented by the EER model, is shown in Figure A.15.

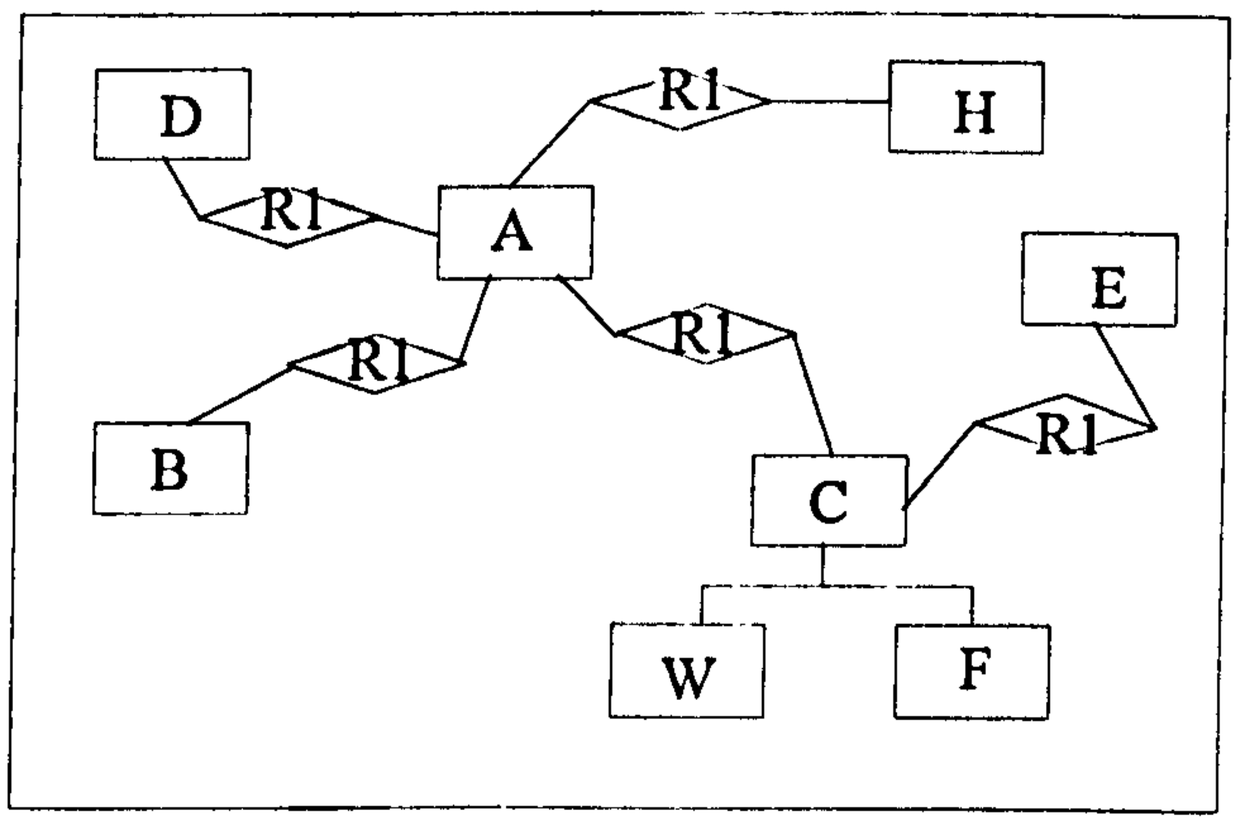


Figure A.15 : An example of the EER schema in case 1

While the third part of the index pack is not empty (i.e. there exist newly activated, confirmed relation(s) of the non-deleted concepts), this rule will be used to mark the relevant case(s), which includes (include) not only the maximum finding element but also the non-deleted concepts (relations) in the finding set, by means of the overlapped links of the working case.

G-Rule 4.2.5 : If the overlapped cases of the working case include some finding elements, but not the maximum finding element in the finding set,
then the overlapped cases, which include the finding elements of the working case, are marked.

An example of the general rule is depicted in Figure A.16.

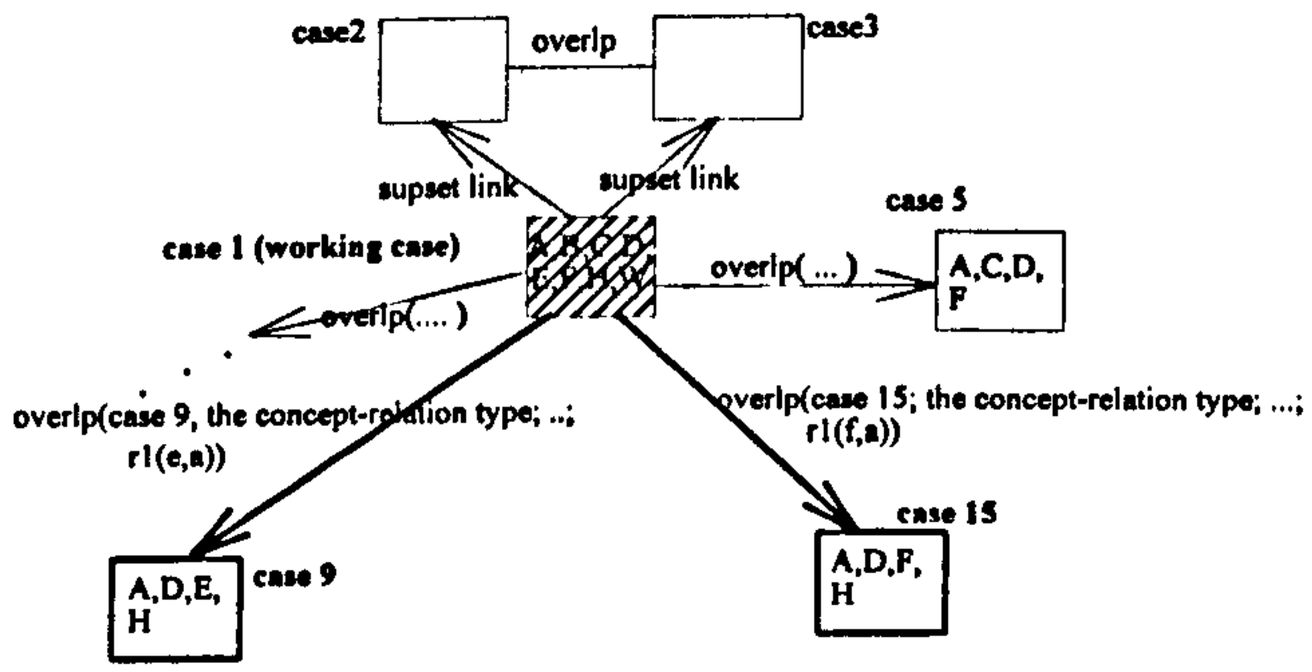


Figure A.16 : An example of G-Rule 4.2.5

The general view of the main body of the working case (case 1) represented by the EER model, is shown in Figure A.17.

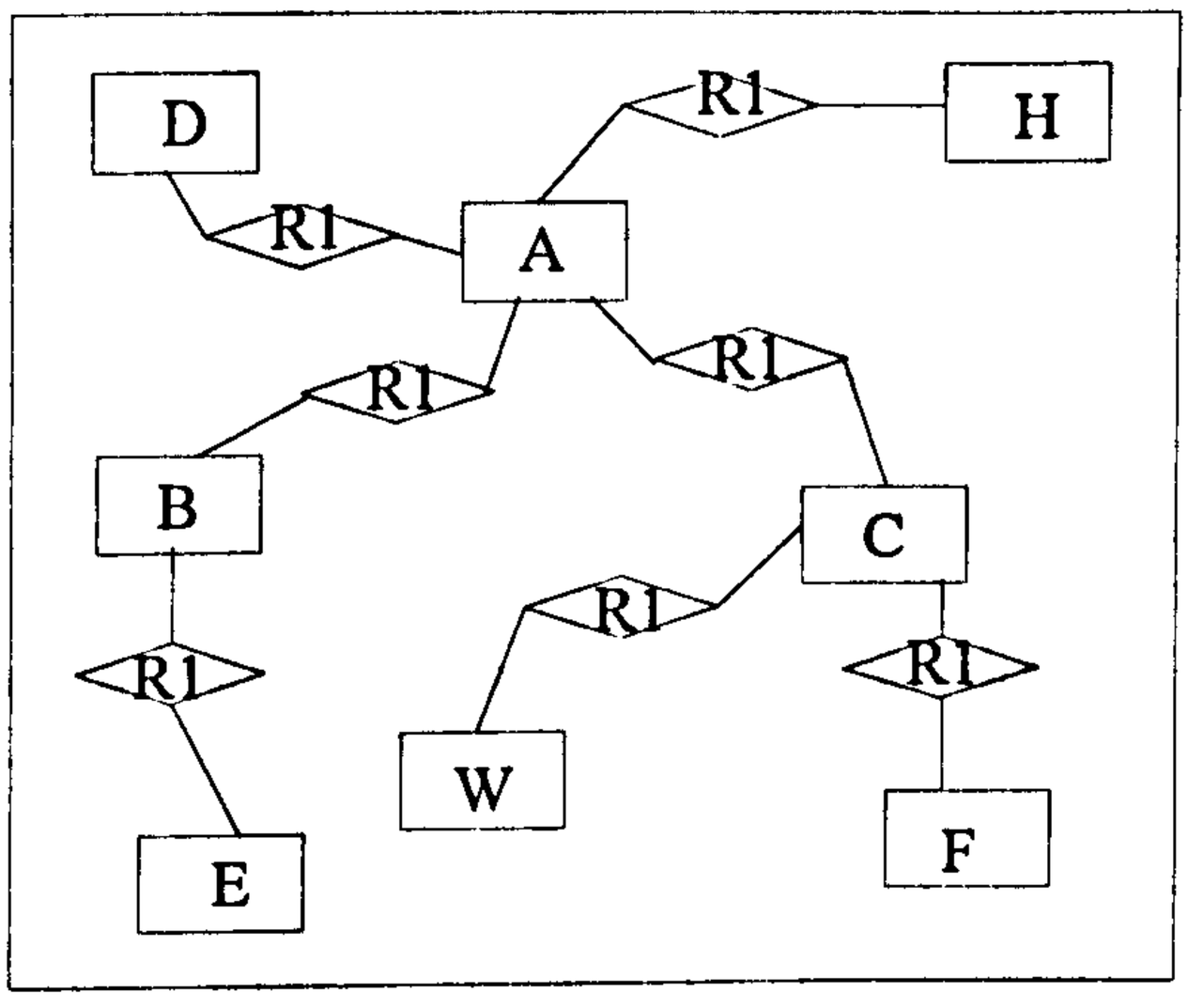


Figure A.17 : An example of the EER schema in case 1

While the third part of the index pack is not empty (i.e. there exist newly activated, confirmed relation(s) of the non-deleted concepts), this rule will be used to mark the relevant case(s) including some finding elements[70] by means of the overlapped links of the working case.

A.3 The meta-knowledge of the third part of the key recalling process: the adapt sub-process

The third type of control knowledge is for the adaptation sub-process.

Adaptation Rules: The meta knowledge used here is for adapting the relationships and attributes of the entities in the initial conceptual data schema.

---

[70] This situation is applied to whether the contents of the overlapped cases of the working case together cover the maximum finding element or not

After the relevant entities have been checked to see whether they fit the needs of the user requirements, two general adaptation rules are available to carry out the adaptation of properties and contents of the relevant entities. The first is for the functional roles of the entities (i.e. coverage properties of the internal relation: generalisation; and degree, cardinality and attributes of the external relation). The second rule is for the contents of the entities (i.e. the attributes of the entity).

G-Rule 6.1 : If a relevant relationship of entities in the initial conceptual data schema is clicked,

then (1) the corresponding concepts are highlighted by means of the has-exemplar links; (2) the same relevant entities in the overlapped cases (the concept-relation type) of the working case/selected case(s) are highlighted in the order of decreasing overlapped degree, or/and in the supset-cases (subset-cases) of the working case/selected case(s) are highlighted by travelling up (down) the supset (subset) links, by means of the exemplar links of the highlighted concepts in the concept base; (3) the relationships of the relevant entities in different cases will be activated in turn for comparison with the clicked relationship in the working case/selected cases; (4) the varied relationship of the relevant entities will be displayed as clues to let users examine whether they need to be considered in the current situation (Repeat step 3).

The general process of the adaptation of the relationships in the initial conceptual data schema involving the working case and/or selected cases is depicted in Figure A.18. The assurance of step (1) and (2) in the above general rule is guaranteed by the relevance of the two bases discussed in section 4.2.3.

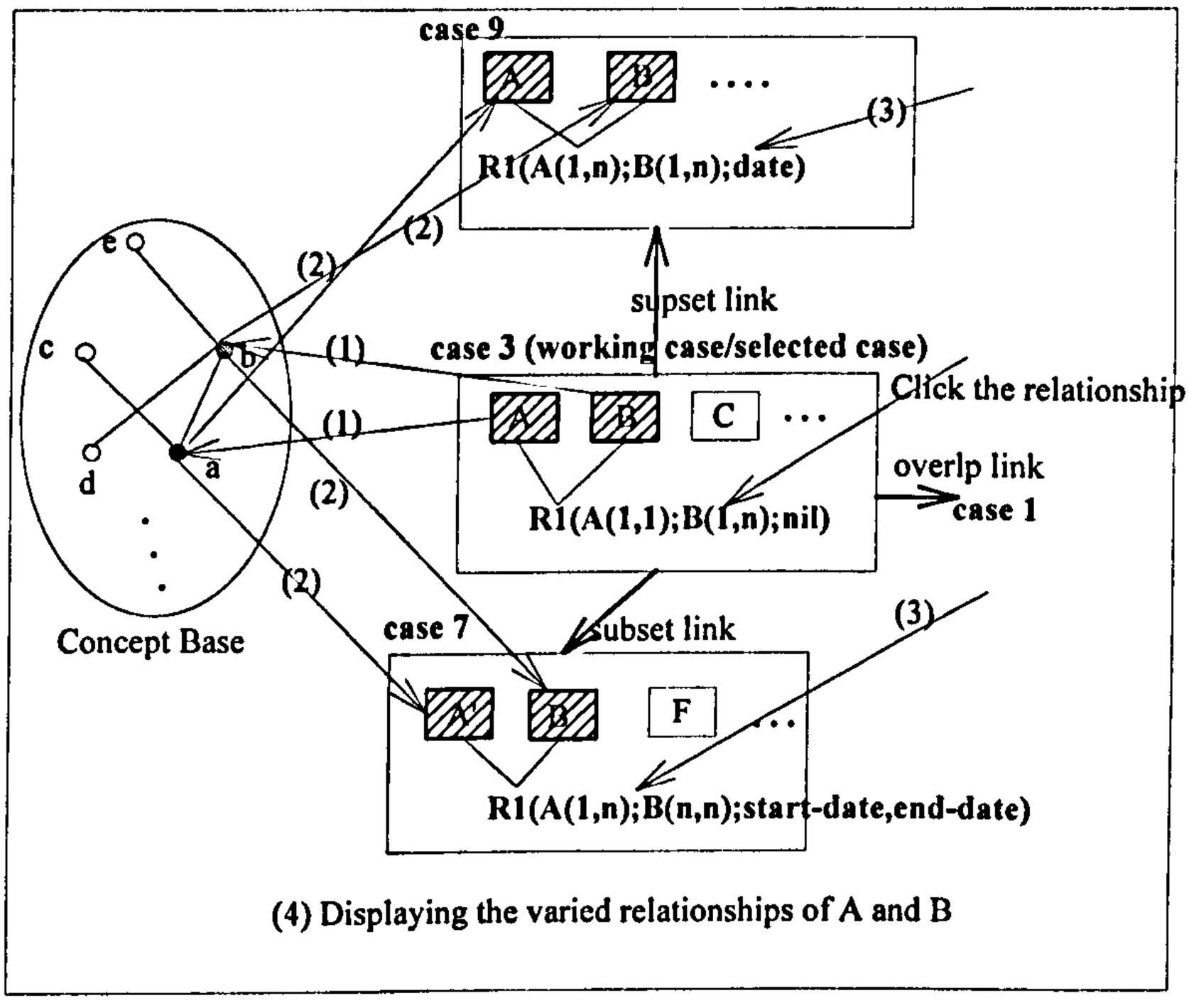


Figure A.18 : The process of the adaptation of the relationship between entities

G-Rule 6.2 : If a relevant entity (including attributes) in the initial conceptual data schema is clicked,

then (1) the corresponding concepts in the concept base are highlighted; (2) by means of the exemplar links of the corresponding concept, the content of entity A in different cases (or stand-alone entitiy) is activated for comparison with the clicked attributes; (3) the varied exemplars of

concept will be displayed as clues to let users examine whether they need to be considered in the current situation (Repeat step 2).

The general process of the adaptation of the attributes of an entity in the initial conceptual data schema involving the working case and/or selected cases is depicted in Figure A.19.

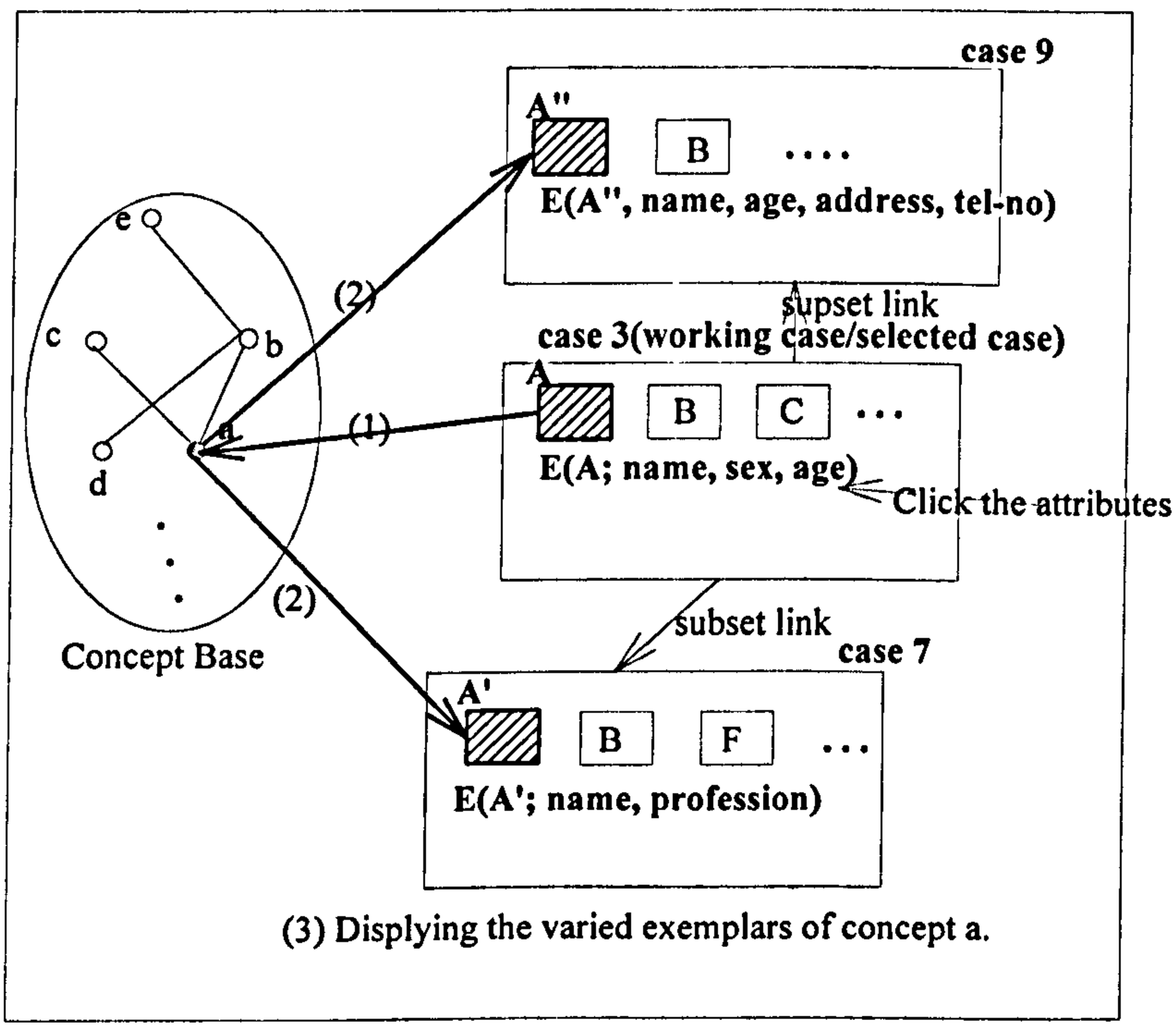


Figure A.19 : The process of the adaptation of the attributes of an entity

Appendix B

The Meta-knowledge of the Store Sub-process

This appendix presents the meta knowledge of the store sub-process, which is concerned with how to store the newly gained knowledge into the DBKM.

B.1 The Finished Schema Involves either a New Concept or a New Relation Between Unrelated Known Concepts

G-Rule 7.1 : If the finished schema is stored as a new case in the case base for reuse,

then (1) the cases that involve the relevant elements (concepts and relations of the new storing finished schema[71]) are located; (2) the subset-super and overlapped links of the new case are created; (3) the overlapped degrees of the new case and its overlapped cases are computed; (4) the new concept nodes and relations or the

---

[71] Because the relations of the cases are from the view of the concept base, the elements of the new stored finished schema are represented in terms of the corresponding concepts and the relations between them by virtue of this circustmance. Thus, the cases in which the rule locates, are determined by whether they include the corresponding relevant concepts and relations of the entities and relationships or not.

new relation(s) between the unrelated known concepts in the concept base are created; (5) the exemplar links of the concepts in the concept base to the relevant entities in the new case are created respectively.

In the above general rule, the third step is, by means of the modified rational model discussed in section 4.3.3, to compute the overlapped degree of the overlapped cases; the fourth step is to create new concept node(s) and relations or the new relation between the unrelated known concepts in the concept base; the fifth step is to create the exemplar links from the relevant concepts in the concept base to the entities of the new case in the case base.

The first and second steps are more complicated, depending on the content of the selected case into which the new entity (concept) is added. The first step is, by using the overlapped and sub-super relations of the selected case(s) and the working case, to locate the cases involving the relevant elements (concepts and relations) of the finished schema. The second step is, by using the outcome of the first step, to create the subset-super and overlapped relations of the new case. When finishing the design work, there are two kinds of selected case(s). The first type is when the selected case(s) is (are) relevant to the working case (i.e. of the subset-cases, or the supset-cases or the overlapped cases of the working case). In this research, the first kind is just called the *selected case(s)*. The second type is that the selected case(s) is (are) not relevant to the working case, but to the other selected cases. In this research, the second kind of selected case is named the

ramifying selected case. According to the type of selected case, the primitive general rules for the first and second step in the G-Rule 7.1 are classified as follows:

B.1.1 The Selected Case That Is Relevant to the Working Case

The primitive general rules are further classified into four circumstances illustrated as follows:

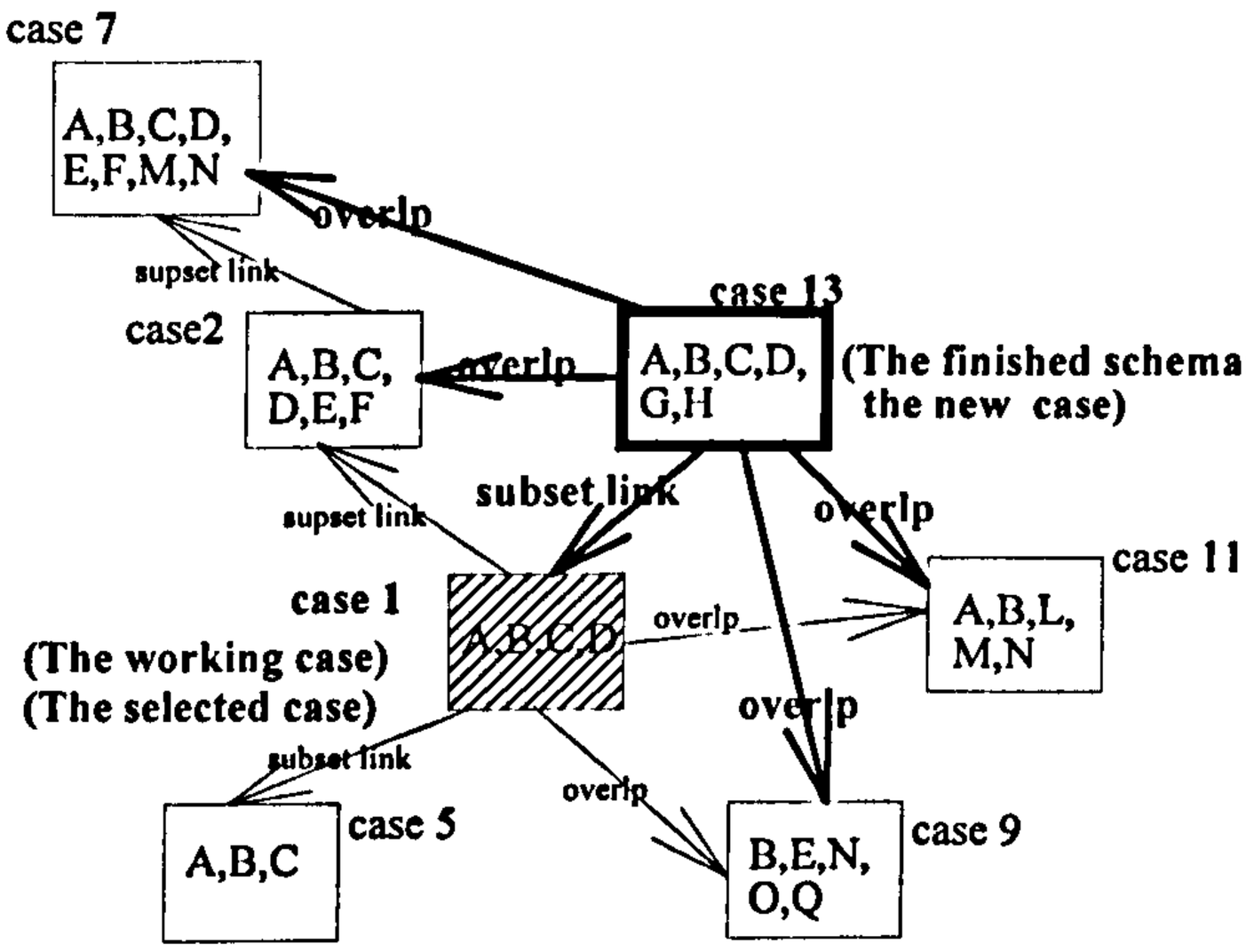
B.1.1.1 Circumstance one: the selected case is the working case

G-Rule 7.1.1.1 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case is the working case[72] and is of the proper subset of the finished schema, then (1) the supset-cases of the selected case are all of the overlapped_cases of the new case; (2) the overlapped cases of the selected case are all of the overlapped case of the new case; (3) the selected case is of the direct subset-case of the new case.

The example of the above general rule is depicted in Figure B.1.

---

[72] According to the definition of the overlapped relation between cases discussed in section 4.2.2.2, the working case can be of the overlapped case of itself.



1. The working case, case 1, is the selected case.

2. The supset-cases of the selected case are case 2 and case 7.
   The overlapped cases of the selected case are case 11 and case 9.

3. The finished schema including the new entities, G and H, for the case base (i.e. the corresponding concepts, g and h) is stored as case 13 in the case base

   The direct subset-case of the new case is case 1
   The overlapped cases of the new case are case 2, case 7, case 9 and case 11

(For simplification, only some links are shown)

Figure B.1 : An example of G-Rule 7.1.1.1

G-Rule 7.1.1.2 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case is the working case[73] and is not of the proper subset of the finished schema,

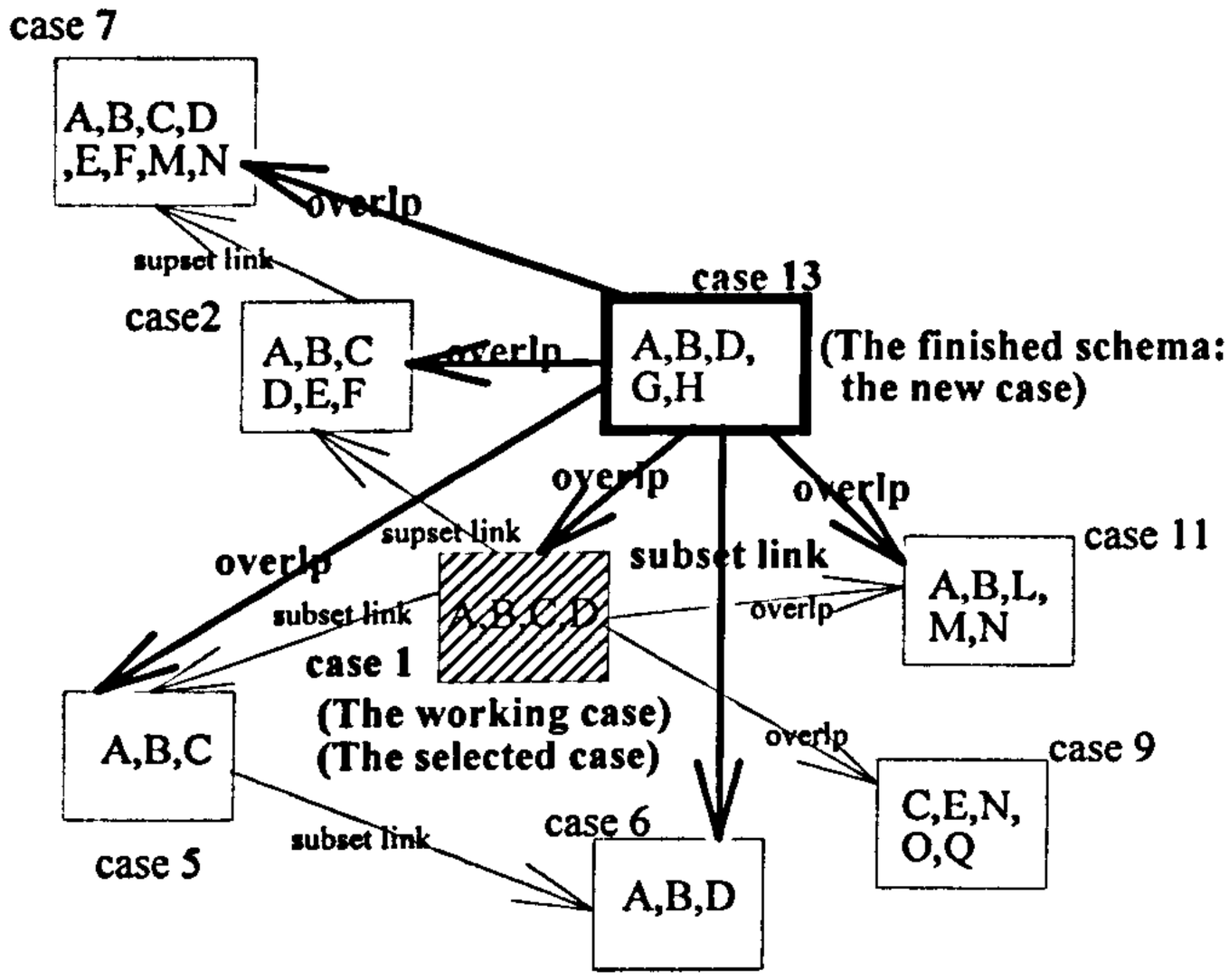
then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of

---

[73] See footnote 72.

the subset-cases of the selected cases depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, the overlapped case must be of the overlapped case of the new case; (4) the selected case is of the overlapped case of the new case.

An example of the above general rule is depicted in Figure B.2. Step (2) involves travelling down the subset link(s) of the selected case to discover whether the subset-cases of the selected case are of the proper subset of the finished schema. This process will proceed continuously until the first encountered case that is of the proper subset of the finished schema or the end of the path. Step (3) is, by means of the overlapped links of the selected case, to identify the overlapped cases that are relevant to the finished schema.



1. The working case, case 1, is the selected case and is not of the proper subset of the finished schema.
2. The supset-cases of the selected case are case 2 and case 7. The subset-cases of the selected case are case 5 and case 6. The overlapped cases of the selected case are case 11 and case 9.
3. Travelling down the subset-cases of the selected case, the first encountered case that is of the proper subset of the finished schema is case 6.
4. Case 9, which is of the overlapped case of the selected case, is not relevant to the finished schema.
5. The finished schema, including the new entities G and H for the case base (i.e. the corresponding concepts, g and h), is stored as case 13 in the case base.
   The direct subset-case of the new case is case 6.
   The overlapped cases of the new case are case 2, case 5, case 7 and case 11.

(For simplification, only some links are shown)

Figure B.2 : An example of G-Rule 7.1.1.2

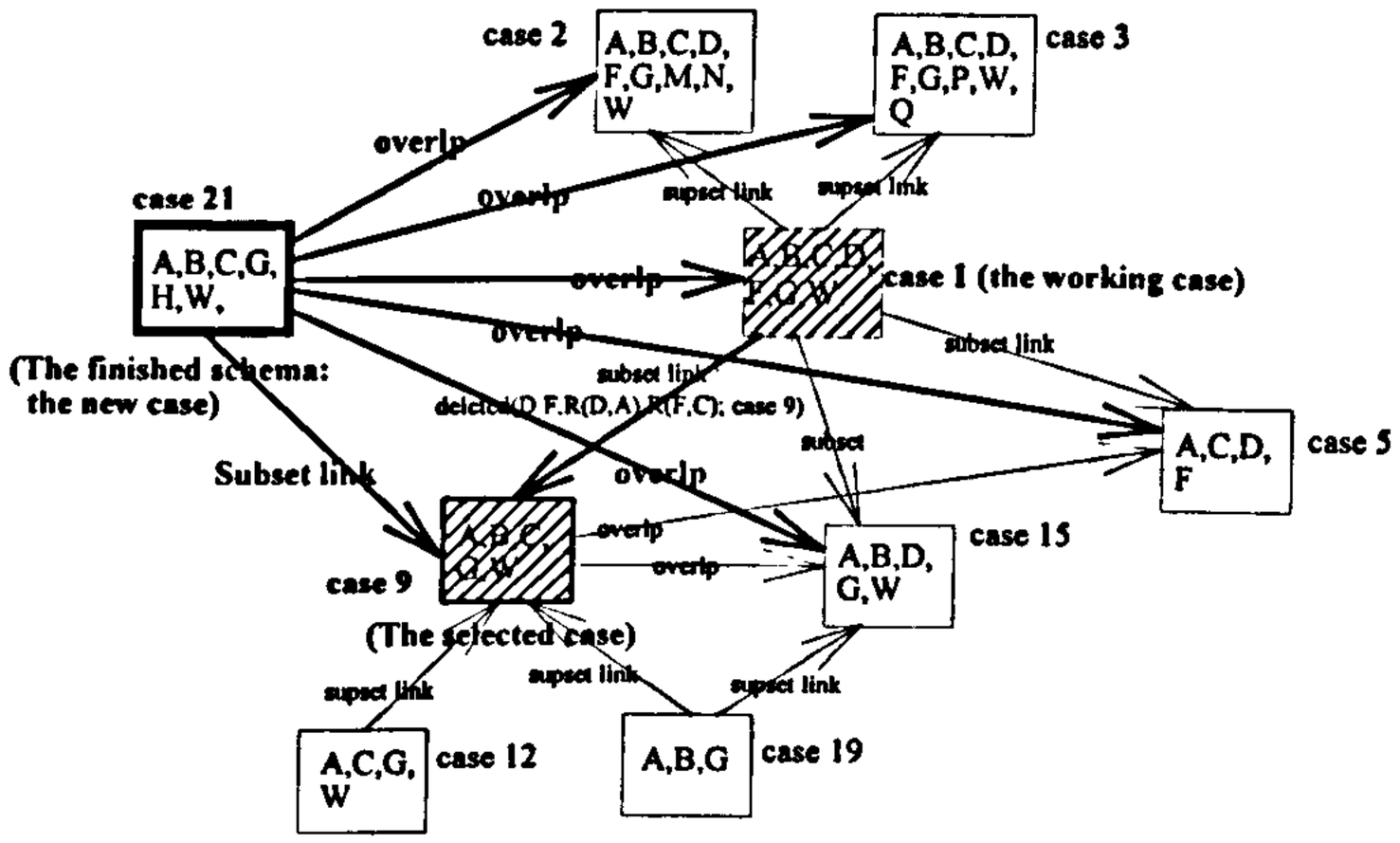
B.1.1.2 Circumstance two: the selected case is the subset-case of the working case

G-Rule 7.1.1.3 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case, which is the subset-case of the working case, no matter whether it

partially or fully satisfies the user requirements (deleted entity), is of the proper subset of the finished schema,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the overlapped cases of the selected case are all of the overlapped case of the new case; (3) the selected case is of the direct subset-case of the new case; (4) the subset-cases of the selected case are of the subset-case of the new case.

The example of the above general rule is depicted in Figure B.3.



1. Case 9 is a selected case, which is of the subset-case of the working case.

2. The supset-cases of the selected case are case 1, case 2 and case 3.
   The subset-cases of the selected case are case 12 and case 19.
   The overlapped cases of the selected case are case 5 and case 15.

3. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 21 in the case base.
   The direct subset-case of the new case is case 9.
   The subset-cases of the new case are case 9, case 12 and case 13.
   The overlapped cases of the new case are case 1, case 2, case 3, case 5 and case 15.

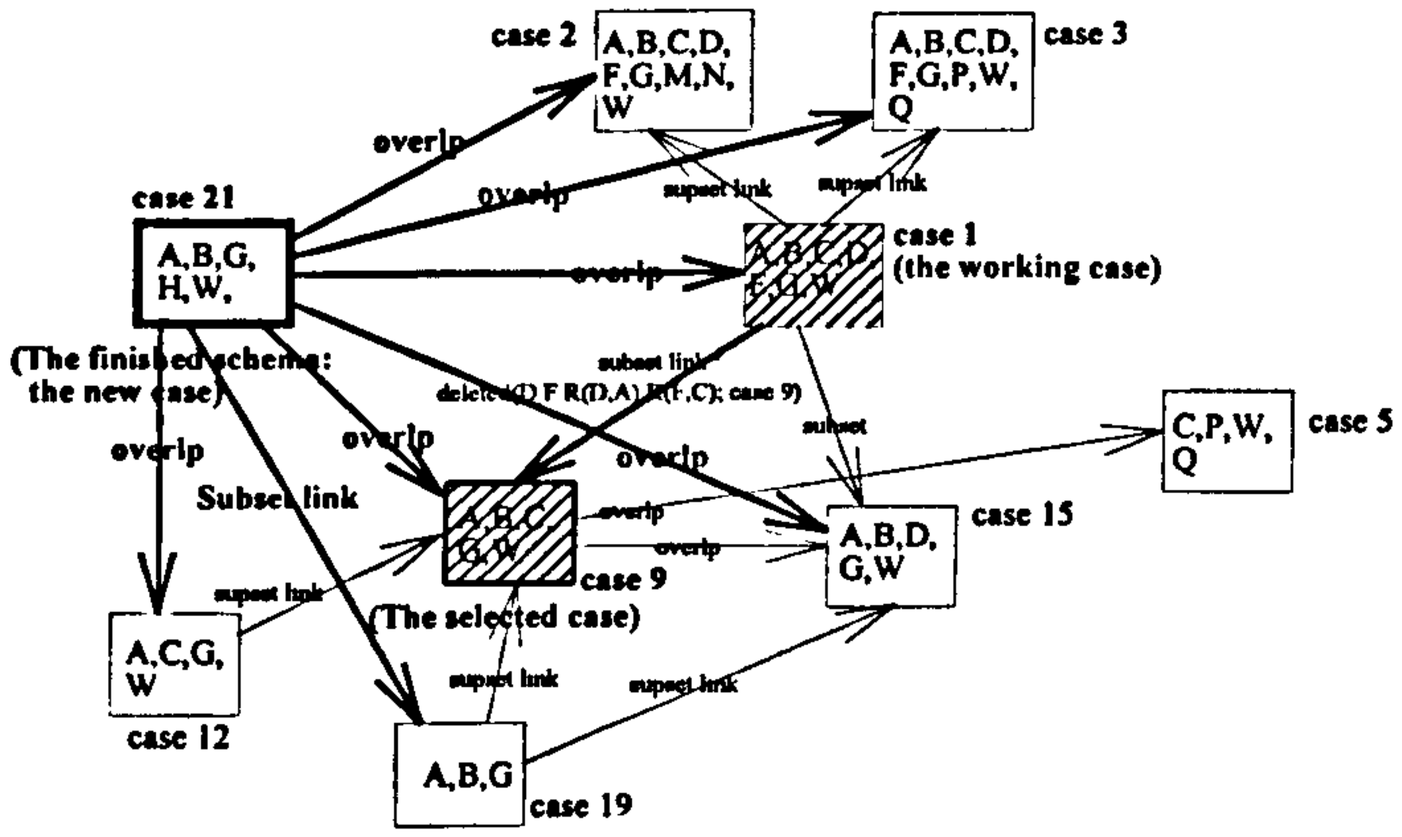
(For simplification, only some links are shown)

Figure B.3 : An example of G-Rule 7.1.1.3

G-Rule 7.1.1.4 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case, which is the subset-case of the working case, no matter whether it partially or fully satisfies the user requirements (deleted entity), is not of the proper subset of the finished schema, then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the selected cases depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, the overlapped case must be of the overlapped case of the new case; (4) the selected case is of the overlapped case of the new case.

An example of the above general rule is depicted in Figure B.4. Step (2) is, by travelling down the subset link(s) of the selected case, to determine whether the subset-cases of the selected case are of the proper subset of the finished schema. This process will proceed continuously until the first encountered case that is of the proper subset of the finished schema or the end of the path. Step (3) is, by

means of the overlapped links of the selected case, to identify the overlapped cases that are relevant to the finished schema.



1. The selected case 9, which is of the subset-case of the working case, is not of the proper subset of the finished case.
2. The supset-cases of the selected case are case 1, case 2 and case 3.
   The subset-cases of the selected case are case 12 and case 19.
   The overlapped cases of the selected case are case 5 and case 15.
3. Travelling down the subset links of the selected case, case 19 is of the proper subset of the finished schema.
4. Case 5, which is of the overlapped case of the selected case, is not relevant to the finished schema.
5. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 21 in the case base.
   The direct subset-case of the new case is case 19.
   The overlapped cases of the new case are case 1, case 2, case 3, case 9, case 12 and case 15.

(For simplification, only some links are shown)

Figure B.4 : An example of G-Rule 7.1.1.4

B.1.1.3 Circumstance three: the selected case is the supset-case of the working case

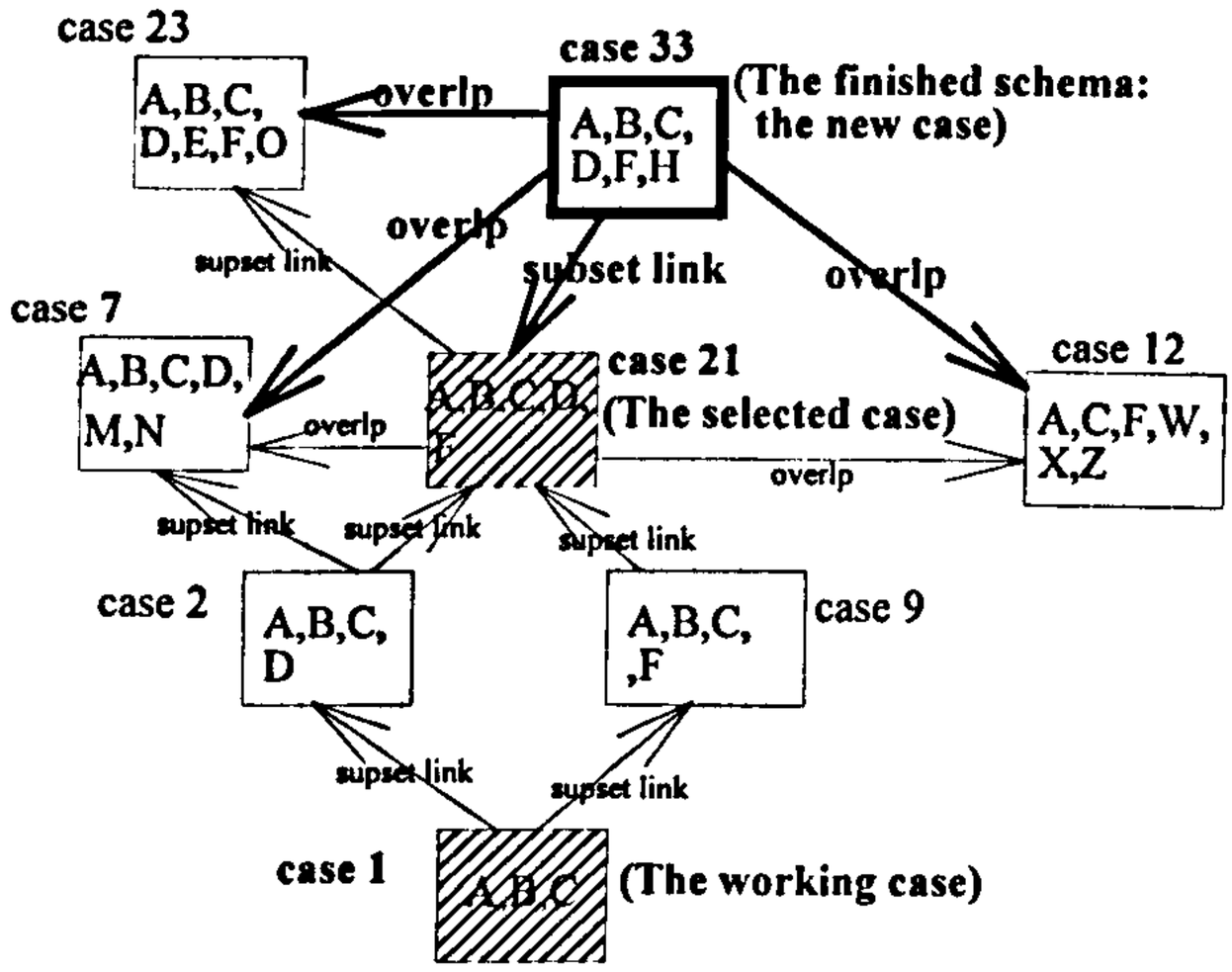
G-Rule 7.1.1.5 : If the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[74], which is a supset-case of the

[74] In this rule, the selected case, including the maximum finding element, is of the supremum (the least upper bound) based on the maximum finding element.

working case, is of the proper subset of the finished schema and includes the maximum finding element,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the overlapped cases of the selected case are all of the overlapped case of the new case; (3) the selected case is of the direct subset case of the new case.

An example of the above general rule is depicted in Figure B.5.



1. Case 1 is the working case.
2. The selected case, case 2, is of the proper subset of the finished schema and includes the maximum finding element {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case is case 23.
   The overlapped cases of the selected case are case 7 and case 12.
4. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-case of the new case is case 21.
   The overlapped cases of the new case are case 7, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.5 : An example of G-Rule 7.1.1.5

G-Rule 7.1.1.6 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[75], which is a supset-case of the working case, is not of the proper subset of the finished schema, but includes the maximum finding element,

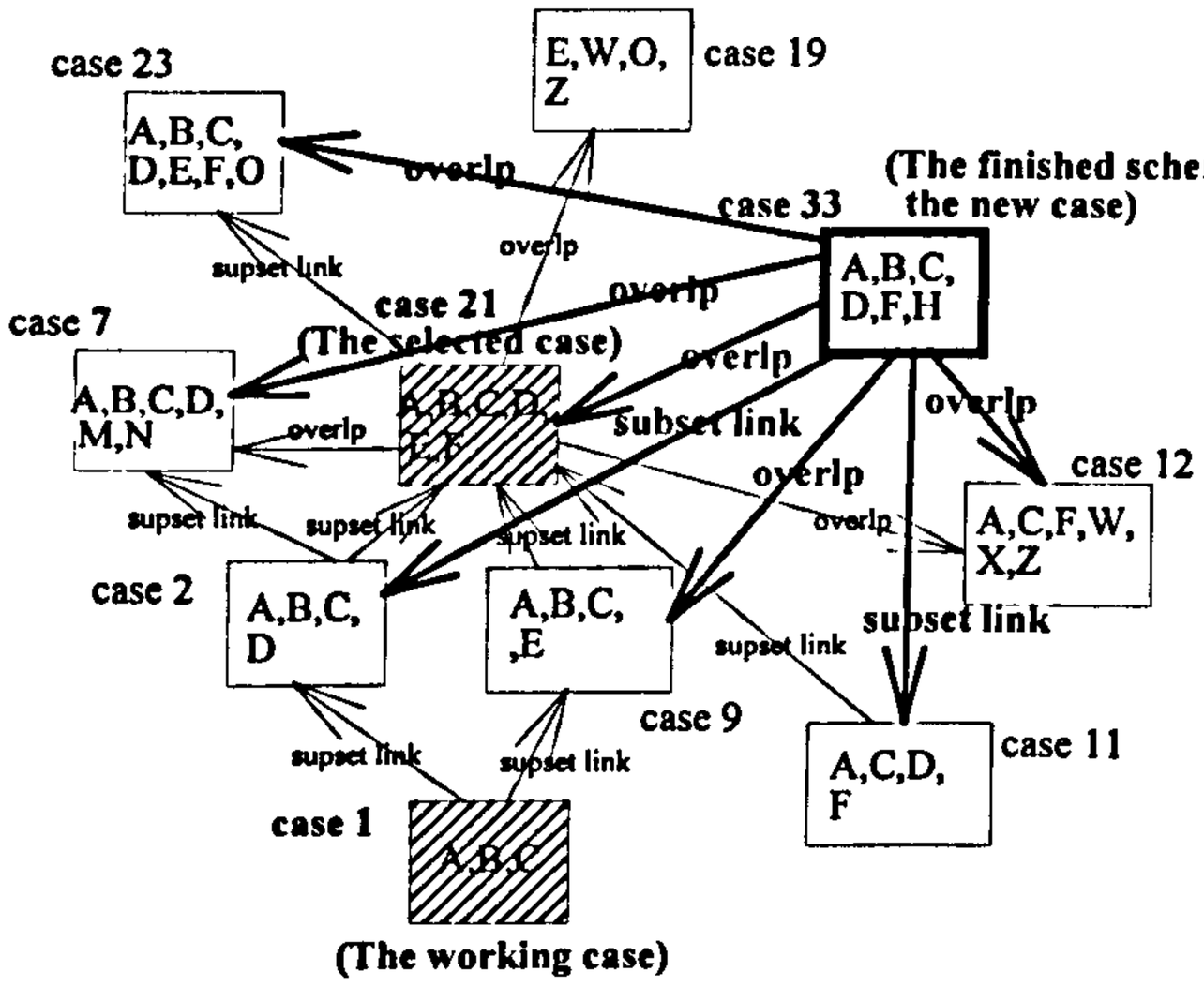
then (1) the supset-cases of the selected case are all of the overlapped case of the working case; (2) the relation types of the subset-cases of the selected cases depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, the overlapped case must be of the overlapped case of the new case; (4) the selected case is of the overlapped case of the new case.

The process of step (2) in the above general rule involves travelling down from the subset link of the selected case until the first case that is of the proper subset of the finished schema or the end case of the path. This process will be carried

---

[75] See footnote 74.

out continuously until all of the subset paths of the selected cases have been checked out. An example of the above general rule is depicted in Figure B.6.



1. Case 1 is the working case.
2. The selected case, case 21, is not of the proper subset of the finished schema, but includes the maximum finding element {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case is case 23.
   The overlapped cases of the selected case are case 7, case 12 and case 19.
4. There are three direct subset-cases of the selected case: case 2, case 9 and case 11.
5. Travelling down from case 2, the first encountered subset-case is of the proper subset of the finished schema: case 2.
   Travelling down from case 9, the first encountered subset-case is of the proper subset of the finished schema: case 1.
   Travelling down from case 11, the first encountered subset-case is of the proper subset of the finished schema: case 11.
6. Case 2 is a direct supset-case of case 1.
7. Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
8. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2 and case 11.
   The overlapped cases of the new case are case 7, case 9, case 12, case 21 and case 23.

(For simplification, only some links are shown)

Figure B.6 : An example of G-Rule 7.1.1.6

G-Rule 7.1.1.7 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the

concept base, and (2) the selected case[76], which is a supset-case of the working case, is of the proper subset of the finished schema and, together with other selected or ramifying cases, covers the maximum finding element, but none of them includes the maximum finding element, then (1) the supset-case of the selected cases are all of the overlapped case of the new case; (2) the relation types of the overlapped cases of the selected case depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the selected case is of the direct subset case of the new case.

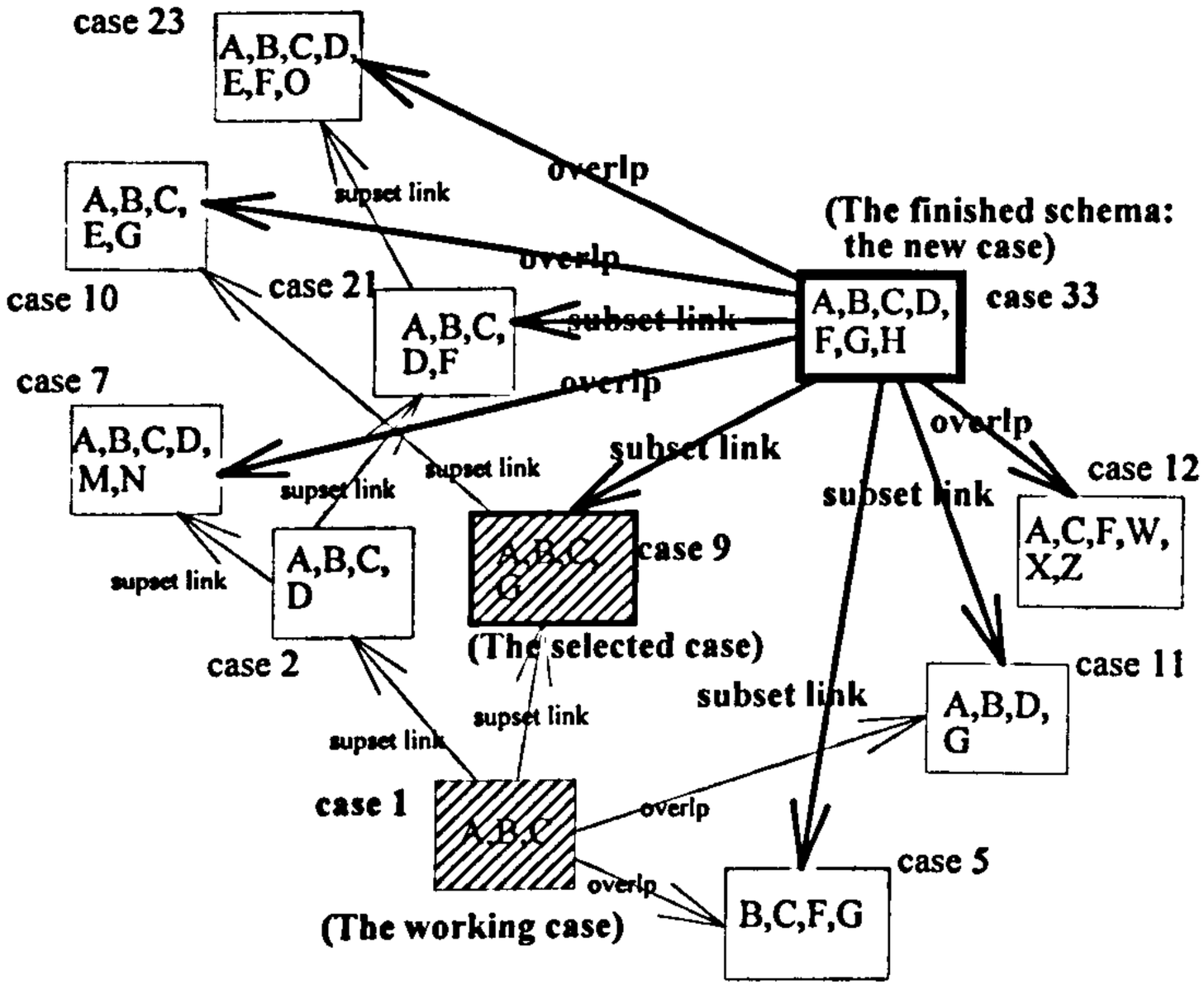
The whole step of the above general rule is carried out systematically by a three-phase process: The first phase is, by means of the supset links of the working case, to find out whether the supset-cases of the working case are of the proper subsets of the finished schema[77]. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phase, aims to find out

---

[76] In this rule, the selected case, including some finding elements but not the maximum finding element, is of the supremum (the least upper bound) based on the finding element.

[77] Obviously, the subset-cases of the working case are all of the proper subsets of the finished schema.

whether the overlapped cases of the selected case are of the overlapped cases of the new case. An example of this rule is depicted in Figure B.7.



1. Case 1 is the working case.
2. The selected case, case 9, is of the proper subset of the finished schema and includes the finding element {G,R(G,B)}.
3. The supset-case of the selected case is case 10.
   The overlapped cases of the selected case 9 are case 2, case 5, case 7, case 11, case 12, case 21and case 23.
4. The overlapped cases of the working case 1 are of the proper subsets of the finished schema case 5 and case 11.
   The supset-cases of the working case 1 are of the proper subsets of the finished schema are case 2 and case 21 (except the selected case 9)
5. Case 2 is of the direct subset-case of case 21.
6. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The selected case 9 is a direct subset-case of the new case.
   Case 5 and case 11, which are of the overlapped cases of the selected case 9, are of the direct-subset cases of the new case.
   Case 21, which is of the overlapped case of the selected case, is of the direct-subset case of the new case.
   Case 10, which is of the supset-cases of the selected case, is of the overlapped case of the new case.
   Case 7, case 12 and 23, which are of the overlapped case of the selected case, are of the overlapped case of the new case.

(For simplification, only some links are shown)

Figure B.7 : An example of G-Rule 7.1.1.7

G-Rule 7.1.1.8 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[78], which is a supset-case of the working case, is not of the proper subset of the finished schema and, together with other selected or ramifying cases, covers the maximum finding element[79], but none of them includes the maximum finding element,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the selected case depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if it is of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the selected case is of the overlapped case of the new case.

---

[78] See footnote 76.

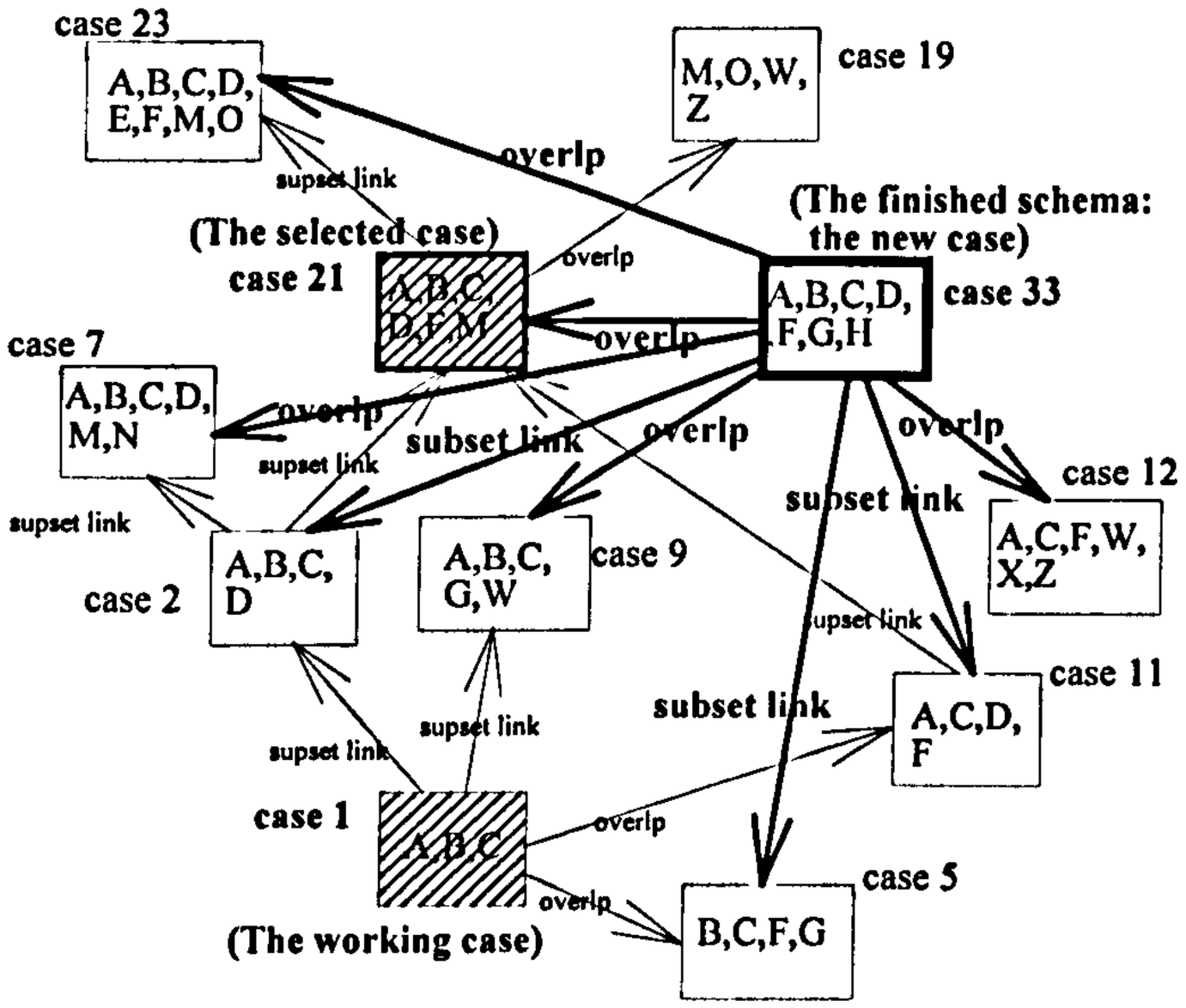
[79] In this rule, the selected case can not be of the super set of the finished schema. Thus, the selected case must be of the overlapped case of the new case

The process of step (2) in the above general rule involves travelling down the subset link of the selected case until the first case that is of the proper subset of the finished schema or the working case if the working case is of the proper subset of the finished schema[80], or by travelling down the subset link of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[81]. This process will be carried out continuously until all of the subset paths of the selected cases (or the working case) have been checked out. Step (3) is carried out in two phases: The first phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of working case are of the proper subset of the finished schema. The second phase is, by using the outcome of the first phase, to find out whether the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is depicted in Figure B.8.

---

[80] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[81] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 21, is not of the proper subset of the finished schema and includes the finding element {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case 21 is case 23.
   The direct subset-cases of the selected case are case 2 and case 11.
   The overlapped cases of the selected case 21 are case 5, case 7, case 9, case 12 and case 19.
4. The first encountered subset-cases of the selected case 21 are of the proper subsets of the finished schema: case 2 and case 11.
5. Case 5, case 11 and case 12, which are the overlapped cases of the working case, are the proper subsets of the finished schema.
   Case 19, which is the overlapped case of the selected case, is not relevant to the finished schema.
6. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   Case 23, which is the direct supset-case of the selected case, is of the overlapped case of the new case.
   Case 2 and case 11, which are of the subset-cases of the selected case, are of the direct subset-case of the new case.
   Case 5, which is of the overlapped case of the selected case, is of the direct subset-case of the new case.
   Case 12, which is of the overlapped case of the selected case, is of the overlapped case of the new case.

(For simplification, only some links are shown)

Figure B.8 : An example of G-Rule 7.1.1.8

B.1.1.4 Circumstance four: the selected case is the overlapped case of the working case

G-Rule 7.1.1.9 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[82], which is the overlapped case of the working case, is of the proper subset of the finished schema and includes the maximum finding element,

then (1) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if it is of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the super-set cases of the selected case are all of the

---

[82] In this rule, the selected case is not the working case (according to the definition of the overlapped relation, the working case can be regarded as an overlapped case of itself), and the overlapped degree of the selected case of the working case, based on the maximum finding element, is above some threshold value.

overlapped case of the new case; (4) the subset-cases of the selected case are all subset-cases of the new case; (5) the relation types of the overlapped cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the direct subset-case of the new case.

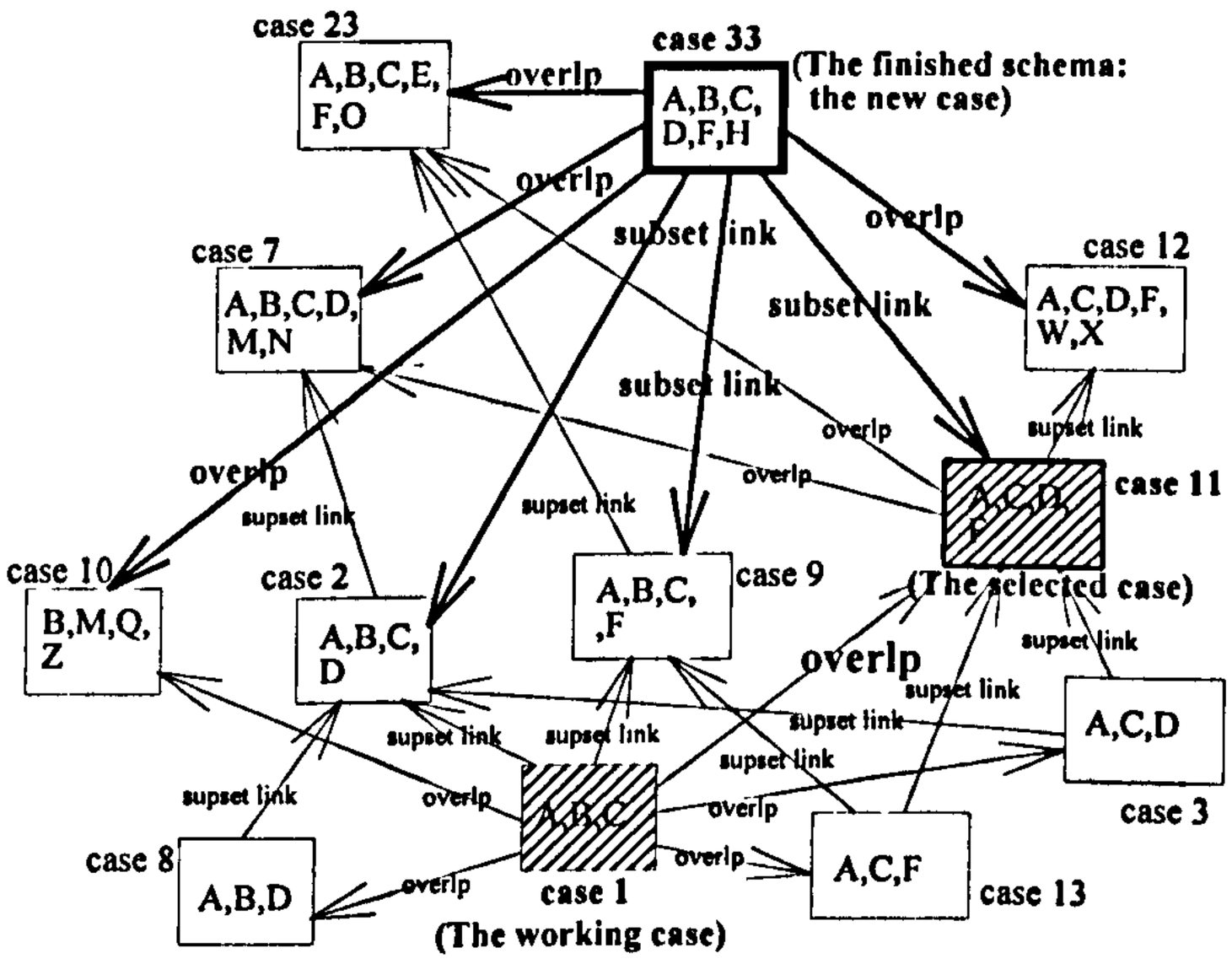
Step (1) to step (6) in the above general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[83], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[84]. This process will be carried out continuously until all of the supset paths or the subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third

---

[83] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[84] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.

phase, by using the outcome of the first and second phase, aims to find out whether the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is depicted in Figure B.9.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 11, is of the overlapped case of the working case and of the proper subset of the finished schema that includes the maximum finding element: {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case is case 12.
   The subset-cases of the selected case are case 3 and case 13.
   The overlapped cases of the selected case are case 2, case 7, case 8, case 9 and case 23.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases (except the selected case) of the working case case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-cases of the working case are of the proper subsets of the finished schema, and are case2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 3, case 8, case 13.
6. Case 3 and case 13 are of the subset-cases of the selected case (case 11).
   Case 8 is of the direct subset-cases of case 2.
7. The finished schema, including the new entity H for the case base (i.e. corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case2, case 9 and case 11.
   The overlapped case of the new storing case are case 7, case 10, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.9 : An example of G-Rule 7.1.1.9

G-Rule 7.1.1.10 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[85], which is the overlapped case of the working case, is not of the proper subset of the finished schema and includes the maximum finding element,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the subset-cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished

---

[85] See footnote 82.

schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (5) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the overlapped case of the new case.

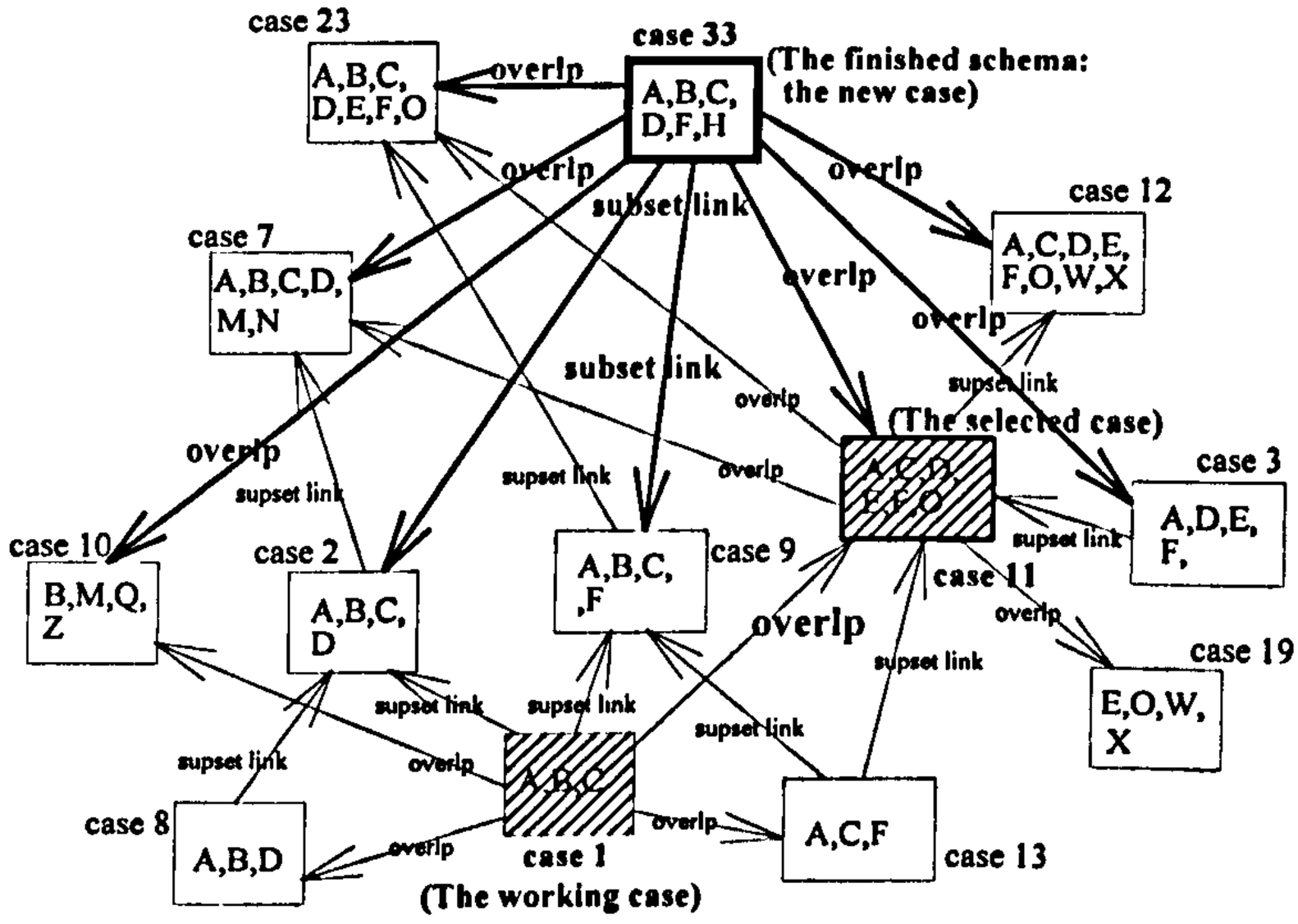
Step (2) to step (6) in the above general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[86], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[87]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome

---

[86] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[87] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.

of the first and second phase, aims to find out whether the subset-cases and the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is depicted in Figure B.10.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 11, is of the overlapped case of the working case and is not of the proper subset of the finished schema that includes the maximum finding element {D,F,R(D,A),R(F,C).
3. The supset-case of the selected case is case 12.
   The subset-cases of the selected case are case 3 and case 13.
   The overlapped cases of the selected case are case 2, case 7, case 8, case 9, case 19 and case 23.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases (except the selected case) of the working case are case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-cases of the working case are of the proper subsets of the finished schema, and are case 2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 8 and case 13.
6. Case 13 is of the subset-case of case 9.
   Case 8 is of the direct subset-case of case 2.
   Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
7. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2 and case 9.
   The overlapped cases of the new case are case 3, case 7, case 10, case 11, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.10 : An example of G-Rule 7.1.1.10

G-Rule 7.1.1.11 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[88], which is of the overlapped case of the working case, is of the proper subset of the finished schema, and together with other selected or ramifying selected cases, covers the maximum finding element, but none of them includes the maximum finding element,

then (1) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the supset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the

---

[88] See footnote 82.

case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the overlapped cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (5) the selected case, if none of the subset-case of the new case is of the supset-case of the selected case, is of the direct subset-case of the new case.

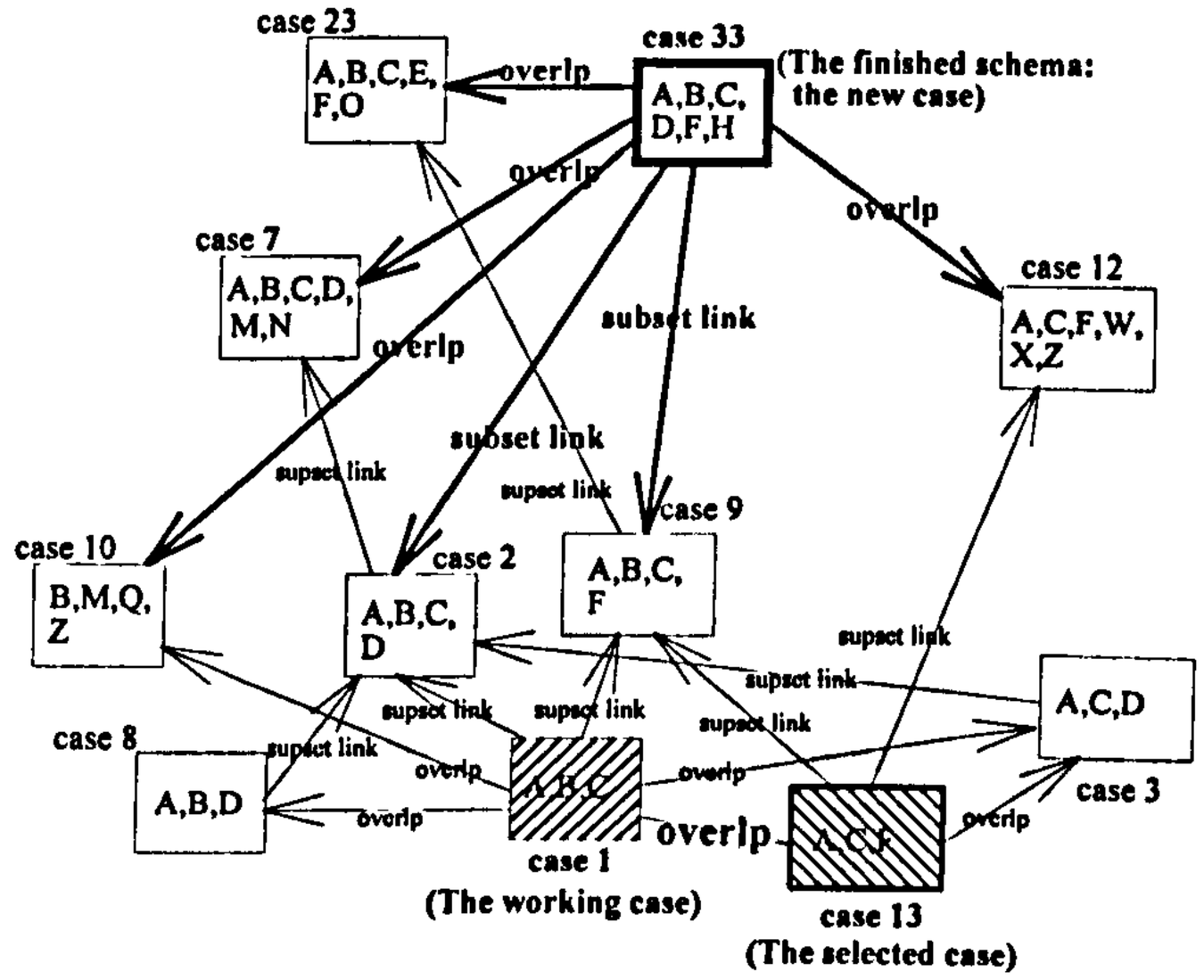
Step (1) to step (5) in the above general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[89], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[90]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the

---

[89] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[90] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.

working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the supset-cases and the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is illustrated in Figure B.11.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 13, is of the overlapped case of the working case, being of the proper subsets of the finished schema and including some finding element: {F,R(F,C)}.
3. The supset-cases of the selected case 13 are case 9, case 12 and case 23.
   The overlapped cases of the selected case 13 are case 1, case 2, case 3, case 7 and case 8.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases of the working case are case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-case of the working case is of the proper subset of the finished schema: case 2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema: case 3, case 8, case 13.
6. Case 3 and case 8 are the subset-cases of case 2.
   Case 13 is the subset-case of case 9.
7. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2 and case 9.
   The overlapped cases of the new case are case 7, case 10, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.11 : An example of G-Rule 7.1.1.11

G-Rule 7.1.1.12 : If (2) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the selected case[91], which is of the overlapped cases of the working case, is not of the proper subset of the finished schema and, together with other selected or ramifying selected cases, covers the maximum finding element, but none of them includes the maximum finding element,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the supset-cases and subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the subset-cases of the selected case depend

---

[91] See footnote 79.

upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (5) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the overlapped case of the new case.

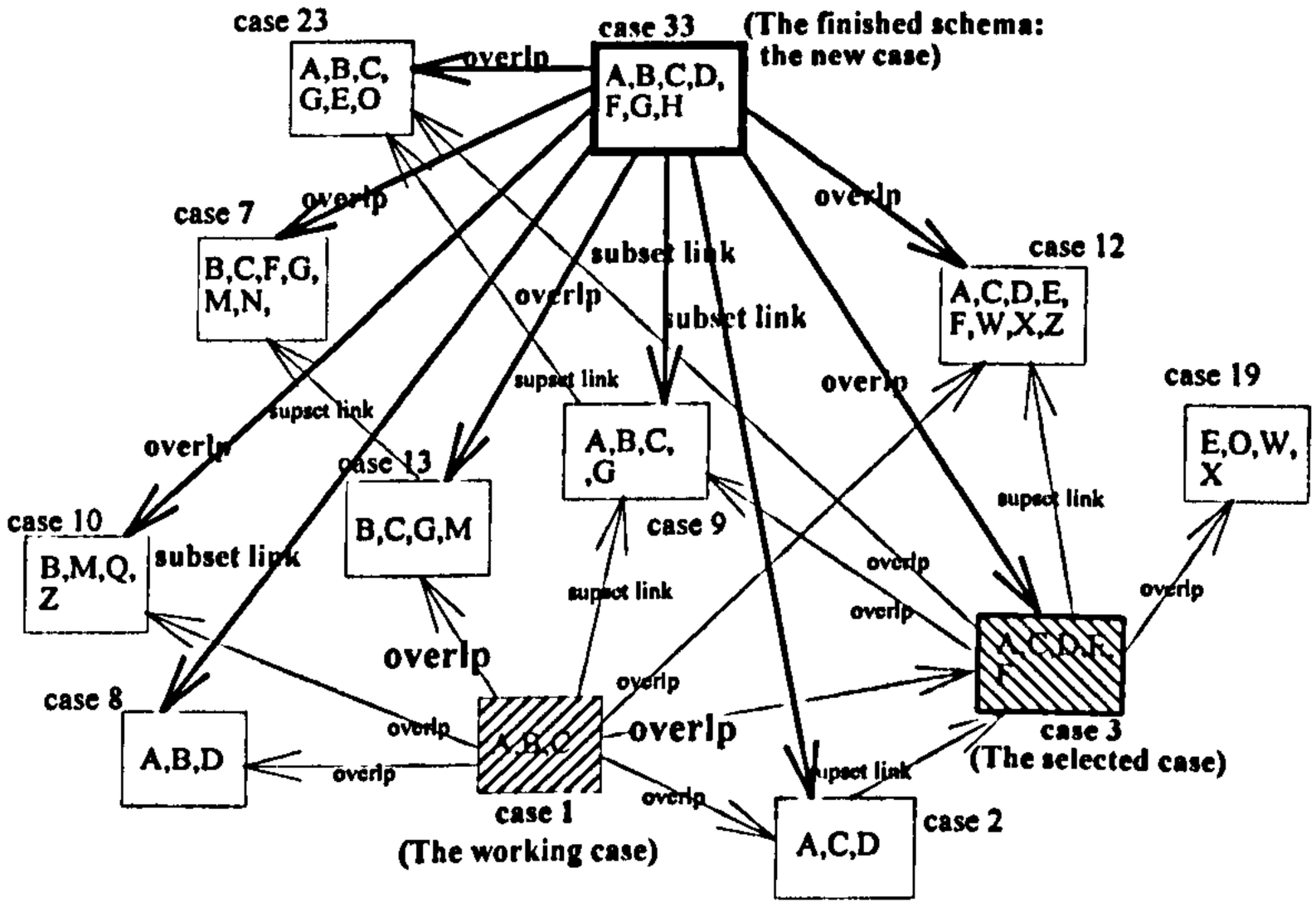
Step (2) to step (6) in the above general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[92], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[93]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the

---

[92] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[93] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.

working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the subset-cases and the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is illustrated in Figure B.12.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 3, is of the overlapped case of the working case, being not of the proper subsets of the finished schema and including some finding element: {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case 3 is case 12.
   The subset-case of the selected case 3 is case 2.
   The overlapped cases of the selected case 3 are case 7, case 8, case 9, case 13, case 19 and case 23.
4. The supset-cases of the working case are case 9 and case 23.
   The overlapped cases (except the selected cases) of the working case are case 2, case 7, case 8, case 10 and case 12.
5. By means of the supset links of the working case, the supset-case of the working case is of the proper subset of the finished schema: case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema: case 2 and case 8.
6. Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
7. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2, case 8 and case 9.
   The overlapped cases of the new case are case3, case 7, case 10, case 12, case 13 and case 23.

(For simplification, omly some links are shown)

Figure B.12 : An example of G-Rule 7.1.1.12

B.1.2 The Ramifying Selected Case That is not Relevant to the Working Case

There are two circumstances in this situation according to what type of cases the ramifying selected cases come from. The first circumstance is where the ramifying selected cases are branched from some selected case. The second circumstance is where the ramifying selected cases are branched from some ramifying selected case.

Thus, the *ramifying selected case* in this situation must be either of the overlapped case(s) of the selected cases[94] that are the supset-cases or the overlapped cases of the working case, **or** of the overlapped case(s) of the other ramifying cases[95] that are the overlapped case(s) of the selected case. In order to locate the relevant cases and to create the relation types of these relevant cases for storing a new case, the primitive general rules illustrated below are mixed with the primitive rules illustrated in situation (I) to handle the number of combinations in which the elements of the finished schema might be from the old cases which are in the situation (I) or/and situation (II).

---

[94] If a ramifying selected case is a supset-case of the selected cases, the ramifying selected case will be viewed as the selected case in the storing stage.

[95] If the ramifying selected case is a supset-case of the other ramifying selected case, the super ramifying selected case will be viewed as the ramifying selected case in the storing stage.

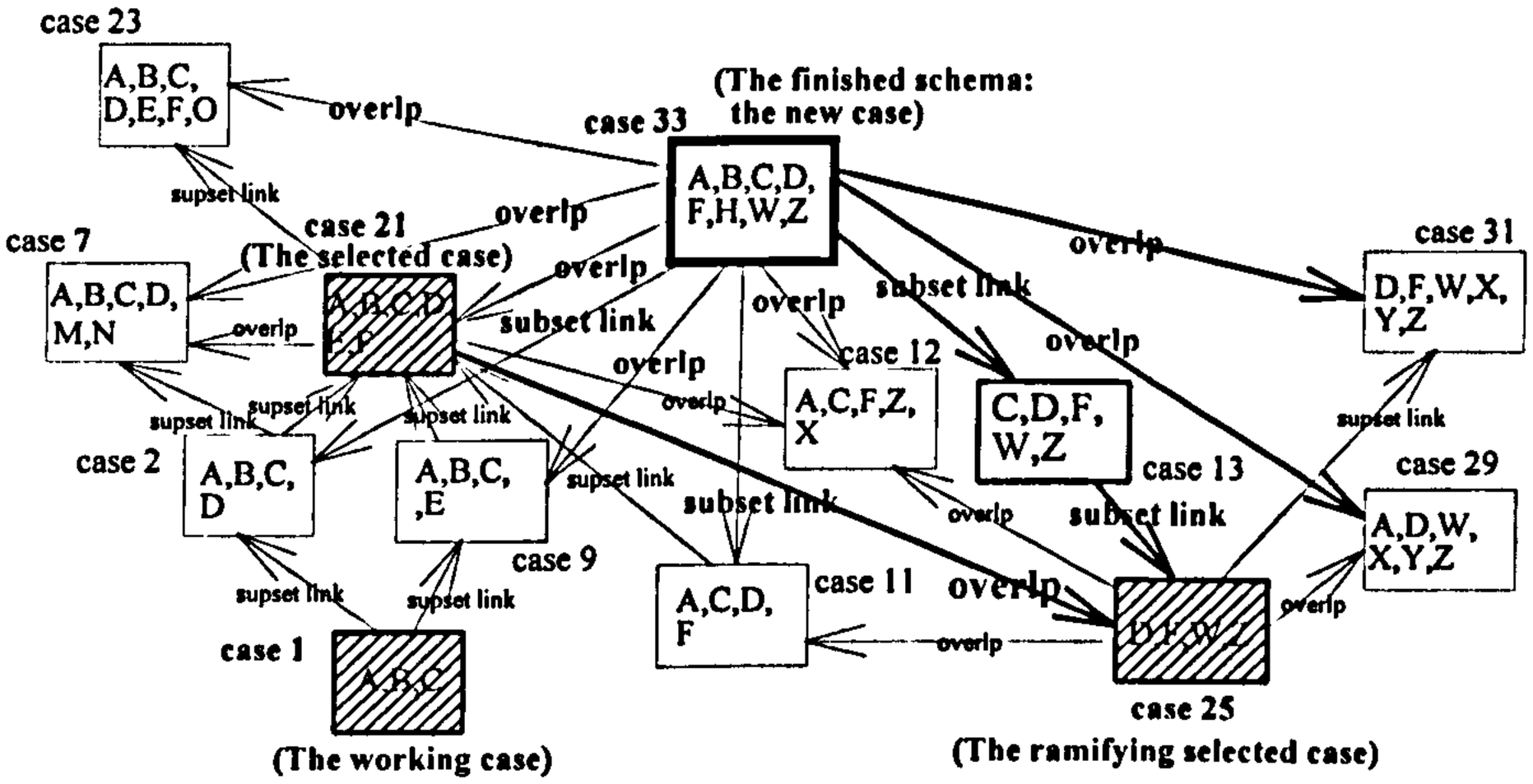
B.1.2.1 Circumstance one: the ramifying selected case is from the selected case

G-Rule 7.1.2.1 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the supset-case of the working case, and is *either* of the subset-case *or* the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the ramifying selected case are all of the subset-case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case, if none of its supset-

cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the general rule is shown in Figure B.13. The whole process is carried out systematically by a two-phase process: The first phase is, by means of G-Rule 7.1.1.7 (the selected case, which is the supset-case of the working case, is of the subset-case of the new case) or G-Rule 7.1.1.8 (the selected case, which is the supset-case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and overlapped cases of the ramifying selected case for the new case.



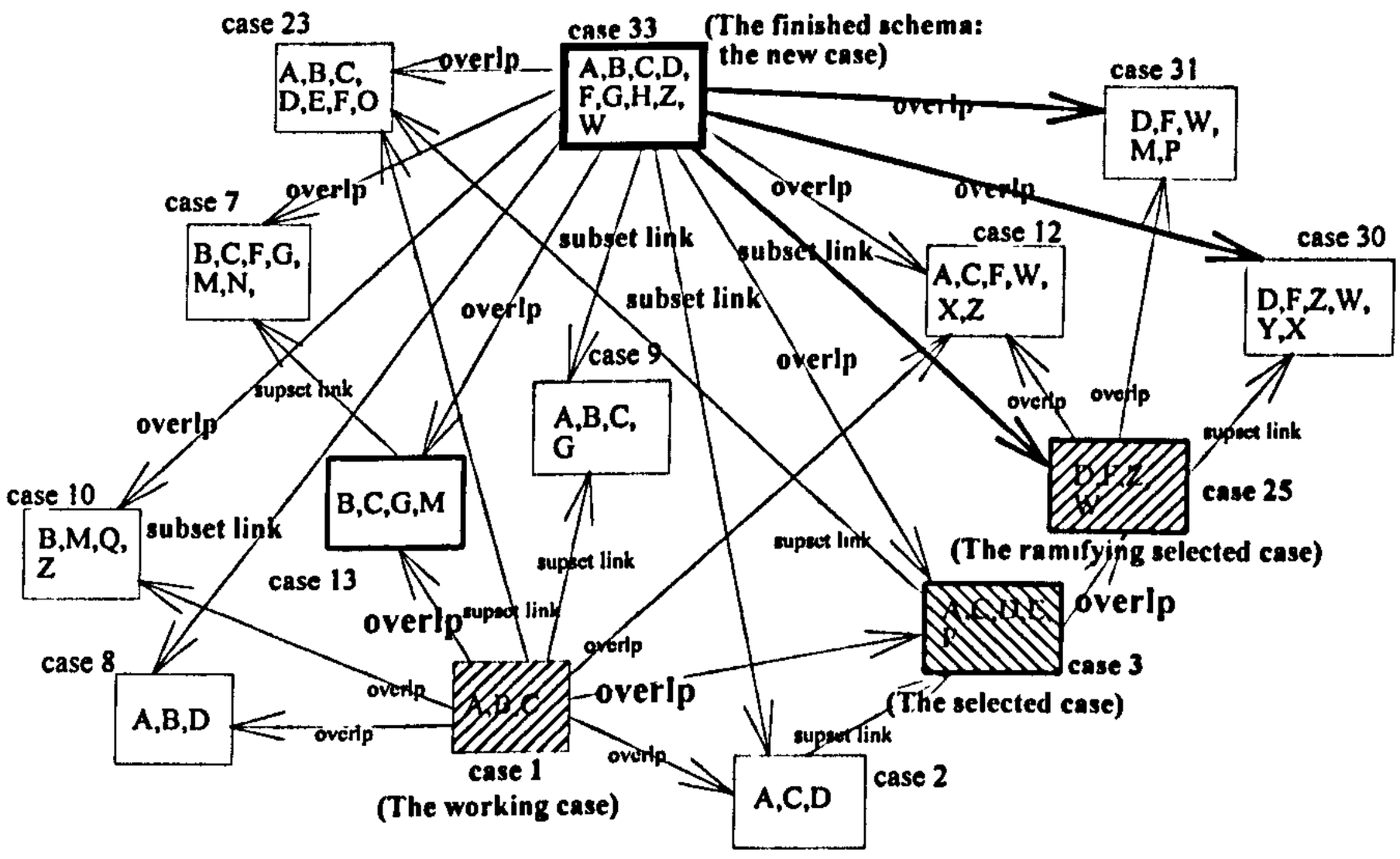
(For simplification, only some links are shown)

Figure B.13 : An example of G-Rule 7.1.2.1

G-rule 7.1.2.2 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the overlapped case of the working case, and is *either* of the subset-case *or* the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the ramifying selected case are all of the subset-case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case, if none of its supset-cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the general rule is shown in Figure B.14. The whole process is carried out systematically by a two-phase process: The first phase is, by means of G-Rule 7.1.1.11 (the selected case, which is the overlapped case of the working case, is of the subset-case of the new case) or G-Rule 7.1.1.12 (the selected case, which is the overlapped case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.14 : An example of G-Rule 7.1.2.2

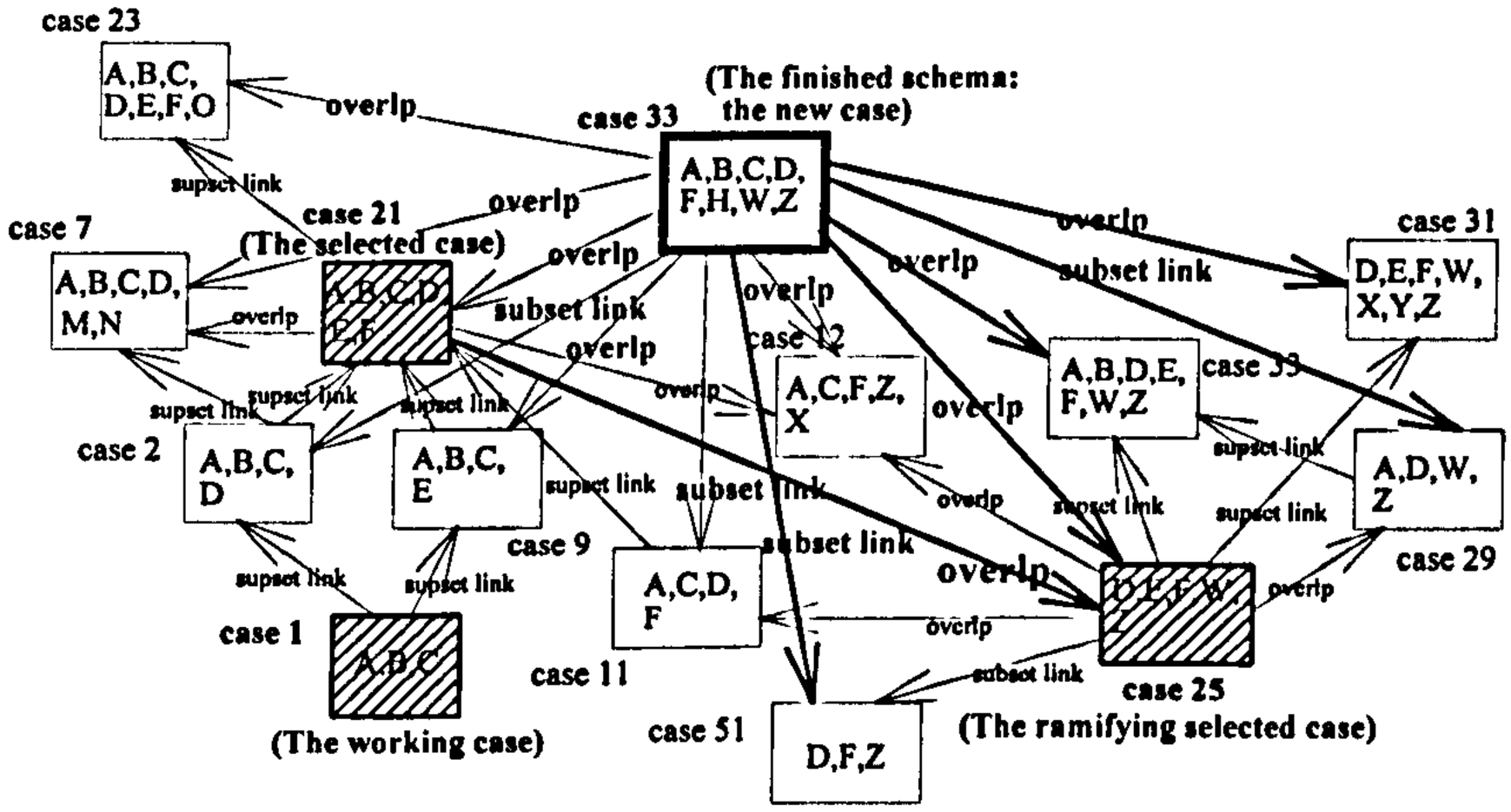
G-Rule 7.1.2.3 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the ramifying selected case, which

is the overlapped case of the selected case, which is of the supset-case of the working case, and is *either* of the subset-case *or* the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case is of the overlapped case of the new case.

An example of the general rule is shown in Figure B.15. The whole process is carried out systematically by a two-phase process. The first phase is, by means of G-Rule 7.1.1.7 (the selected case, which is the supset-case of the working case, is of the subset-case of the new case) or G-Rule 7.1.1.8 (the selected case, which is the supset-case of the working case, is of the overlapped case of the new case), to

check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the subset-cases and the overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

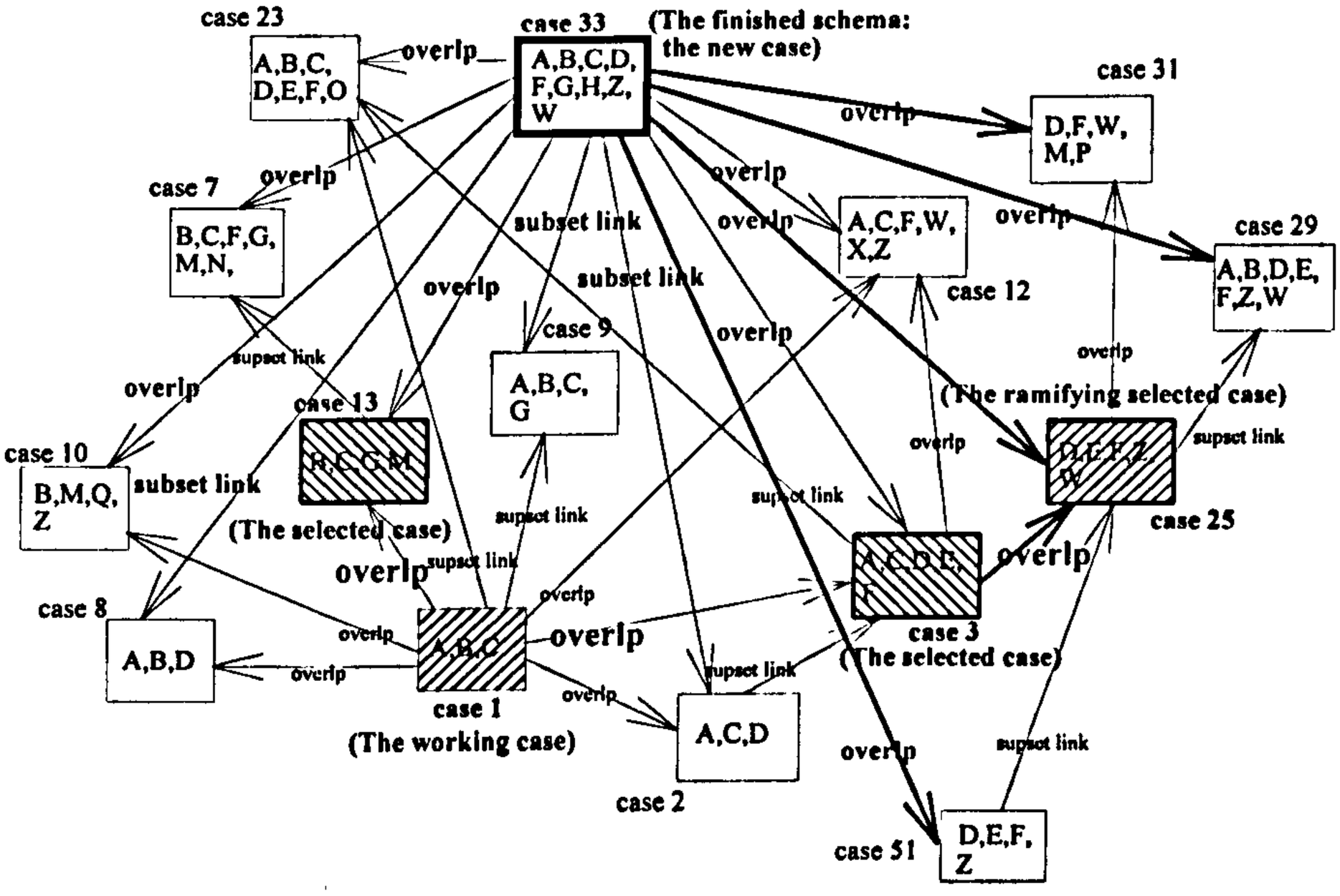
Figure B.15 : An example of G-Rule 7.1.2.3

G-rule 7.1.2.4 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the overlapped case of the working case, and is *either* of the subset-case *or* the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying case are all of the overlapped case of the new case; (2) the relation types

of the subset-cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case is of the overlapped case of the new case.

An example of the general rule is shown in Figure B.16. The whole process is carried out systematically by a two-phase process: The first phase is, by means of G-Rule 7.1.1.11 (the selected case, which is the overlapped case of the working case, is of the subset-case of the new case) or G-Rule 7.1.1.12 (the selected case, which is the overlapped case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the subset-cases and the overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.16 : An example of G-Rule 7.1.2.4

B.1.2.2 Circumstance two: the ramifying selected case is from another ramifying selected case

In this circumstance, the ramifying selected cases are classified into two types. The first type is called the *original ramifying selected case*, abbreviated as *O-ramifying* selected case. The second type is called the *destination ramifying selected case*, abbreviated as *D-ramifying* selected case[96].

G-Rule 7.1.2.5 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the D-ramifying selected case,

---

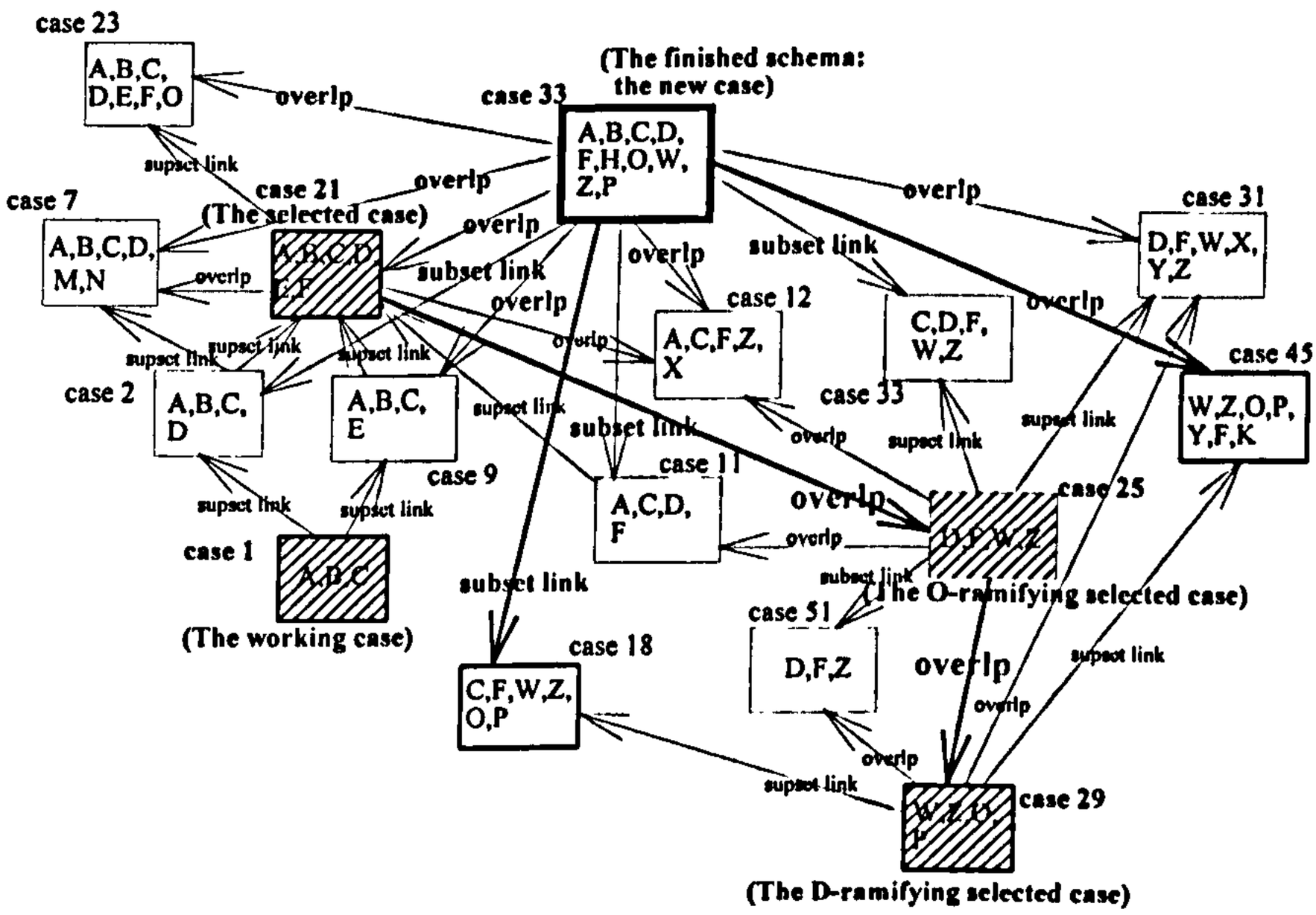
[96] Basically, the O-ramifying selected case can be viewed as a working case of the D-ramifying selected case.

which is the overlapped case of the O-ramifying selected case, which is *either* of the subset-case *or* the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the D-ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the D-ramifying selected case are of the subset-cases of the new case; (3) the relation types of the overlapped cases of the D-ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the D-ramifying selected case, if none of its supset-cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the general rule is shown in Figure B.17. The whole process is carried out systematically by a two-phase process: The first phase is, according to the relation types of the O-ramifying selected case (i.e. the O-ramifying selected case is a subset-case or an overlapped case of the finished schema), to check the

relation types of the relevant cases of the O-ramifying selected case. For example, if the O-ramifying selected case is of the proper subset of the finished schema, the supset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.1.2.1 or G-Rule 7.1.2.2. If the O-ramifying selected case is not of the proper subset of the finished schema, the subset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.1.2.3 or G-Rule 7.1.2.4. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and the overlapped cases of the D-ramifying selected case for the new case.



(For simplification, only some links are shown)

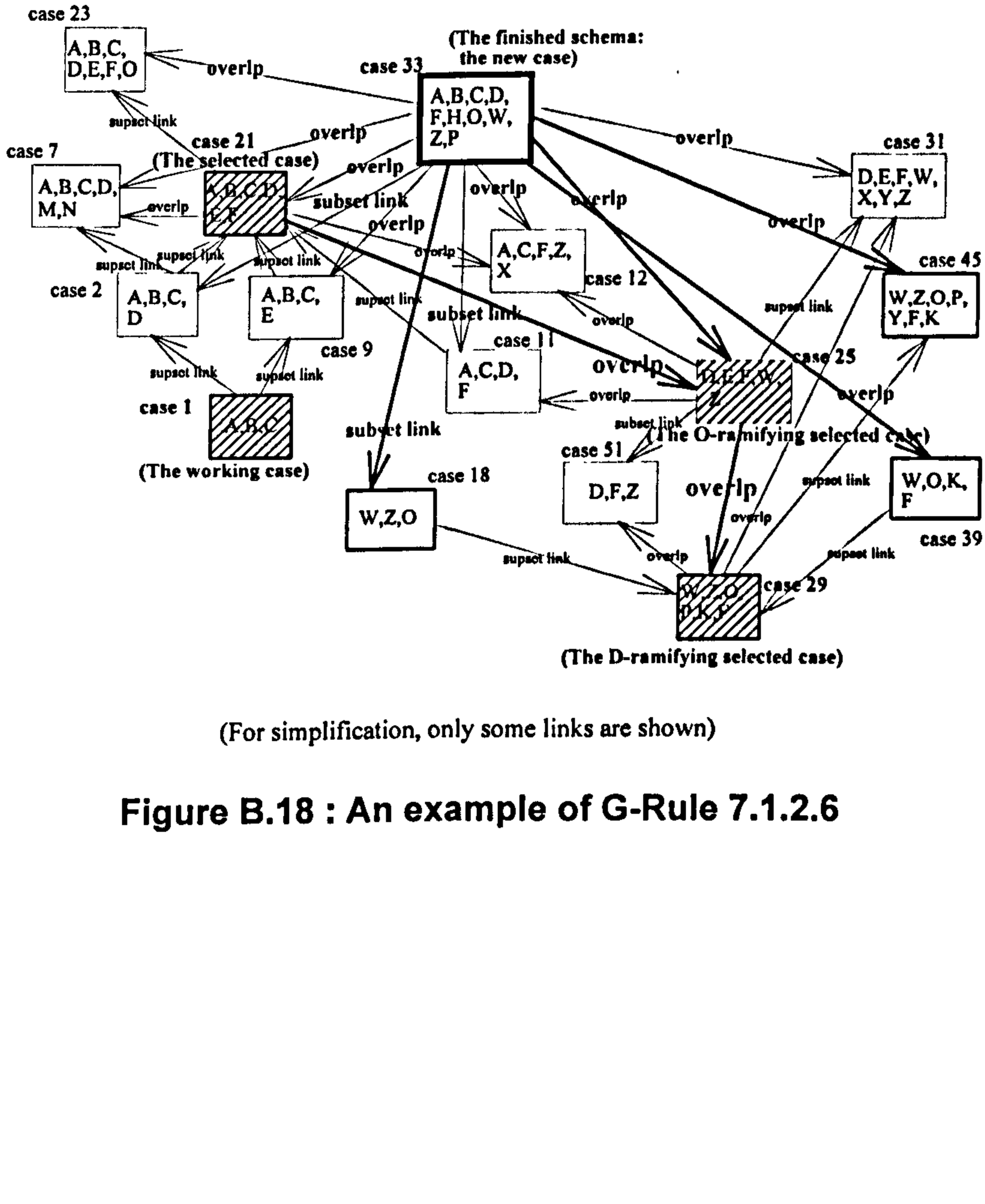
Figure B.17 : An example of G-Rule 7.1.2.5

G-Rule 7.1.2.6 : If (1) the finished schema includes either the new concept(s) or the new relation(s) from the view of the concept base, and (2) the D-ramifying selected case, which is the overlapped case of the O-ramifying selected case, which is *either* of the subset-case *or* the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the D-ramifying case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the D-ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the D-ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the D-ramifying selected case is of the overlapped case of the new case.

An example of the general rule is shown in Figure B.18. The whole process is carried out systematically by a two-phase process: The first phase is, according to

the relation types of the O-ramifying selected case (i.e. the O-ramifying selected case is a subset-case or an overlapped case of the finished schema), to check the relation types of the relevant cases of the O-ramifying selected case. For example, if the O-ramifying selected case is of the proper subset of the finished schema, the supset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.1.2.1 or G-Rule 7.1.2.2 If the O-ramifying selected case is not of the proper subset of the finished schema, the subset-cases and the overlapped cases of the O-ramifying selected case will be checked out by means of G-Rule 7.1.2.3 or G-Rule 7.1.2.4. The second phase is, by using the outcome of the first phase, to find out the relation types of the subset-cases and the overlapped cases of the D-ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.18 : An example of G-Rule 7.1.2.6

B.2 The Finished Schema Involves a New Property of the Relationship Between Known Related Entities (the second situation of the second aspect of influencing the knowledge accumulation)

G-Rule 7.2 : If the finished schema is stored as a new case in the case base for reuse in later sessions (under the second situation of the second aspect of influencing the knowledge accumulation),

then (1) the cases that include the relevant elements (concepts and relations of the new case) are located; (2) the subset-super and overlapped links of the new case are created; (3) the overlapped degrees of the new case and its overlapped cases are computed; (4) the exemplar links of the concepts in the concept base to the relevant entities in the new case are created respectively.

In the above general rule, the third step is, by means of the modified ration model discussed in section 4.3.3, to compute the overlapped degree of the overlapped cases; the fourth step is to create the exemplar links from the relevant concepts in the concept base to the entities in the new case in the case base.

The considerations of the first and second steps of the rule are close to those in G-Rule 7.1. Thus, according to the types of selected case, the primitive general rules for the first and second steps in G-Rule 7.2 are classified as follows:

B.2.1 The Selected Case That is Relevant to the Working Case

The primitive general rules are further classified into four circumstances illustrated as follows:

B.2.1.1 Circumstance one: the selected case is the working case

G-Rule 7.2.1.1 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case is the working case[97] and is of the super-set of the finished schema,

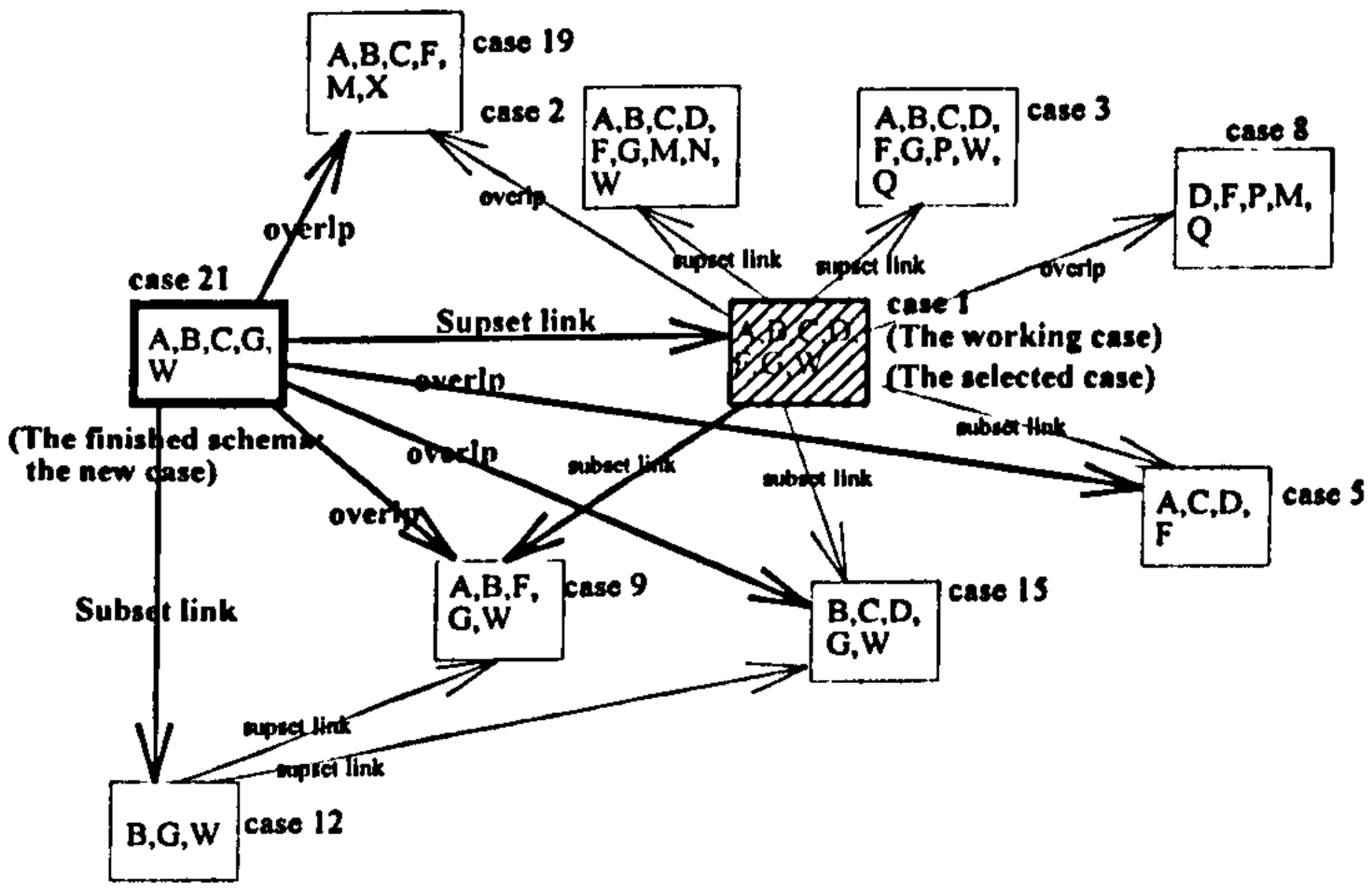
then (1) the selected case is the direct supset-case of the new case; (2) the relation types of the direct subset-cases of the selected case are all of the overlapped case of the new case; (3) the relation types of the non-direct subset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the

---

[97] According to the definition of the overlapped relation between cases discussed in section 4.2.2.2, the working case can be of the overlapped case of itself.

new case; otherwise it is the overlapped case of the new case; (4) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, the overlapped case must be of the overlapped case of the new case.

The process from step (4) to step (5) in the above general rule is carried out systematically in two phases: The first phase involves travelling down from the subset links of the selected case until the first case that is of the proper subset of the finished schema or the end case of the path. The second phase is, by means of the overlapped links of the selected case, to find out the overlapped cases of the selected case are relevant to the finished schema. The example of the above general rule is illustrated in Figure B.19.



1. The working case 1 is the selected case.
2. The supset-cases of the selected case 1 are case 2 and case 3.
   The direct subset-cases of the selected case 1 are case 5, case 9 and case 15.
   The overlapped cases of the selected case 9 are case 8 and case 19.
   The non-direct subset-case of the slected case 1 is case 12.
3. The finished schema, including the new relationship between entity G and entity W, is stored as case 21 in the case base.
4. The selected case 1 is of the supset of the finished schema.
   Case 8, which is of the overlapped cases of the selected case, is not relevant to the finished schema.
   Case 12 is a proper subset of the finished schema and is of the direct-subset case of the selected case 9 and case 15.
5. The direct supset-case of the new case is case 1.
   The supset-cases of the new case are case 2 and case 3.
   The overlapped cases of the new case are case 5, case 9, case 15 and case 19.
   The direct subset-case of the new case is case 12.

(For simplification, only some links are shown)

Figure B.19 : An example of G-Rule 7.2.1.1

B.2.1.2 Circumstance two: the selected case is the subset-case of the working case

G-Rule 7.2.1.2 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[98], which is a subset-case of the working case, is of the

---

[98] In this rule, the selected case, including the non-deleted concepts (relations), is of the infimum (the greatest lower bound), based on some wanted deleting-element (not the maximum deleting element), but still existing in the subset-cases of the working case, i.e. if the selected case and its subset-cases that do not include some non-deleted concepts (relations) form a set, the supset-cases

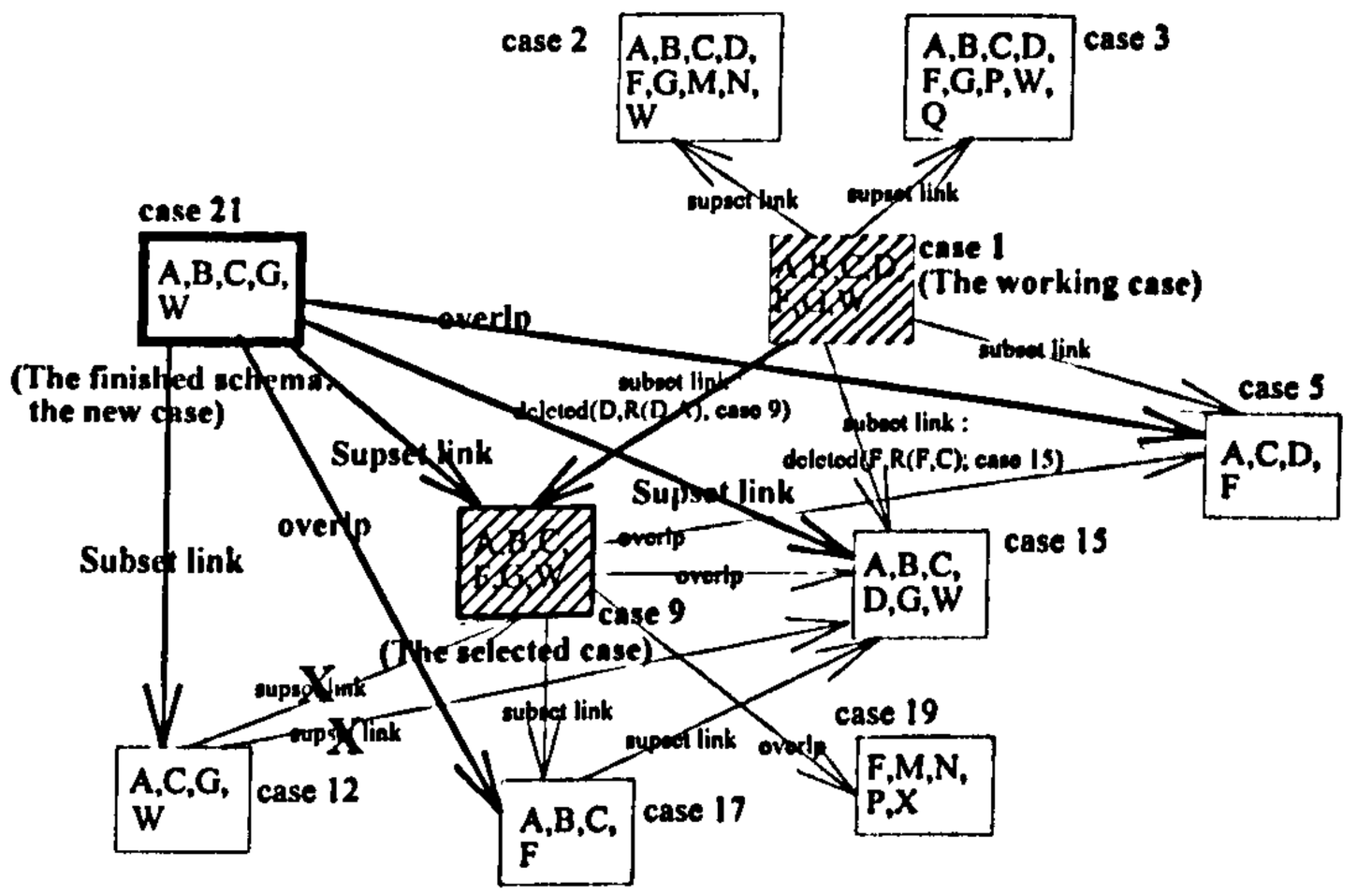
super-set of the finished schema and partially satisfies the user requirement (i.e. the content of the selected case involves some deleting element),

then (1) the selected case is the direct supset-case of the new case; (2) the relation types of the direct subset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of this case are included in the new case, it is a direct subset-case of the new case and the supset link from this case to the selected case will be deleted; otherwise it is the overlapped case of the new case; (3) the relation types of the non-direct subset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the super-set of the finished schema, it is a direct supset-case of the new case; otherwise it is the overlapped case of the new case.

---

of the selected case form a set called B, the set of the union of these two sets is called A, and the selected case is called the infimum of B in A.

The process from step (3) to step (5) in the above general rule is carried out systematically in two phases: The first phase involves travelling down from the subset links of the selected case until the first case that is of the proper subset of the finished schema or the end case of the path. The second phase is, by means of the overlapped links of the selected case, to find out whether the overlapped cases of the selected case that are relevant to the finished schema are of the super-sets of the finished schema. An example of the above general rule is illustrated in Figure B.20.



1. Case 9 is the selected case, which is of the subset-case of the working case, but includes some deleting elements: {F,R(F,C)}.
2. The supset-cases of the selected case 9 are case 1, case 2 and case 3.
   The subset-cases of the selected case 9 are case 12 and case 17.
   The overlapped cases of the selected case 9 are case 5, case 15 and case 19.
3. The finished schema, including the new relationship between entity G and entity W, is stored as case 21 in the case base.
4. The selected case 9 is of the supset of the finished schema.
   Case 15, which is of the overlapped cases of the selected case, is of the super set of the finished schema.
   Case 12 is a proper subset of the finished schema and is of the direct-subset case of the selected case 9 and case 15.
   Case 5, which is of the overlapped case of the selected case, is not a super set of the finished schema.
   Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
5. The direct supset-cases of the new case are case 9 and case 15.
   The supset-cases of the new case are case 9, case 15, case 1, case 2 and case 3.
   The overlapped cases of the new case are case 5 and case 17.
   The direct subset-case of the new case is case 12 (the supset links from case 12 to case 9 and case 15 are deleted).

(For simplification, only some links are shown)

Figure B.20 : An example of G-Rule 7.2.1.2

B.2.1.3 Circumstance three: the selected case is the supset-case of the working case

G-Rule 7.2.1.3 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[99], which is a supset-case of the working case, is of the super-set of the finished schema, but includes the maximum finding element,

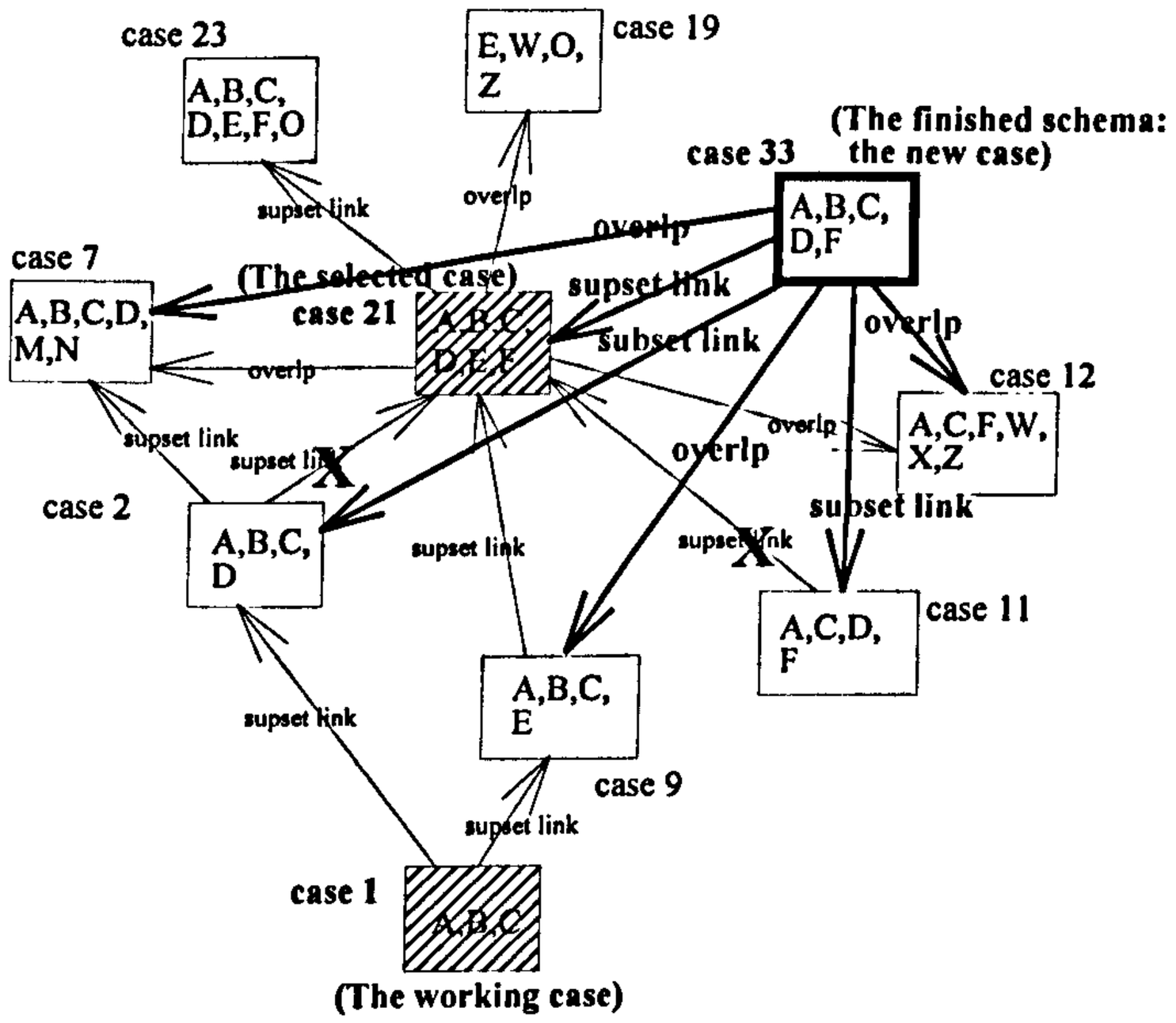
then (1) the selected case is of the direct supset-case of the new case; (2) the relation types of the direct subset-cases of the selected cases depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a direct subset-case of the new case and the supset link from this case to the selected case will be deleted; otherwise it is the overlapped case of the new case; (3) the relation types of the non-direct subset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the overlapped cases of the selected case depend upon whether these cases are

---

[99] See footnote 74.

relevant to the finished schema. If so, the overlapped case must be of the overlapped case of the new case.

The process from step (3) to step (5) in the above general rule is carried out systematically in two phases: The first phase involves travelling down from the subset links of the selected case until the first case that is of the proper subset of the finished schema or the end case of the path. The second phase is, by means of the overlapped links of the selected case, to find out whether the overlapped cases of the selected case are relevant to the finished schema. An example of the above general rule is illustrated in Figure B.21.



1. Case 1 is the working case.
2. The selected case, case 21, is the super set of the finished schema, and includes the maximum finding element {D,F,R(D,A),R(F,C).
3. The supset-case of the selected case is case 23.
   The direct subset-cases of the selected case are case 2, case 9 and case11.
   The non direct subset-case of the selected case is case 1, which is the direct subset-case of the case 2.
   The overlapped cases of the selected case are case 7, case 12 and case 19.
4. The finished schema, including the new relationship between entity D and entity A, is stored as case 33 in the case base.
5. The selected case 21 is of the direct supset-case of the new case.
   Case 2 and case 11, which are the direct subset-cases of the selected case, are of the proper subsets of the finished schema.
   Case 9, which is the direct subset-case of the selected case, is not of the proper subset of the finished schema.
   Case 1, which is the non-direct subset case of the slected case, is of the proper subset of the finished schema.
   Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
6. The direct supset-case of the new case is case 21.
   The direct subset-cases of the new case are case 2 and case 11 (the supset links form case 2 and case 11 to case 21 respectively are deleted).
   The overlapped case of the new case are case 7, case 9 and case 12.

(For simplification, only some links are shown)

Figure B.21 : An example of G-Rule 7.2.1.3

G-Rule 7.2.1.4 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[100], which is a supset-cases of the working case, is of the proper subset of the finished schema and, together with other selected or ramifying cases, covers the maximum finding element, but none of them includes the maximum finding element,

then (1) the supset-case of the selected cases are all of the overlapped case of the new case; (2) the relation types of the overlapped cases of the selected case depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the selected case is of the direct subset case of the new case.

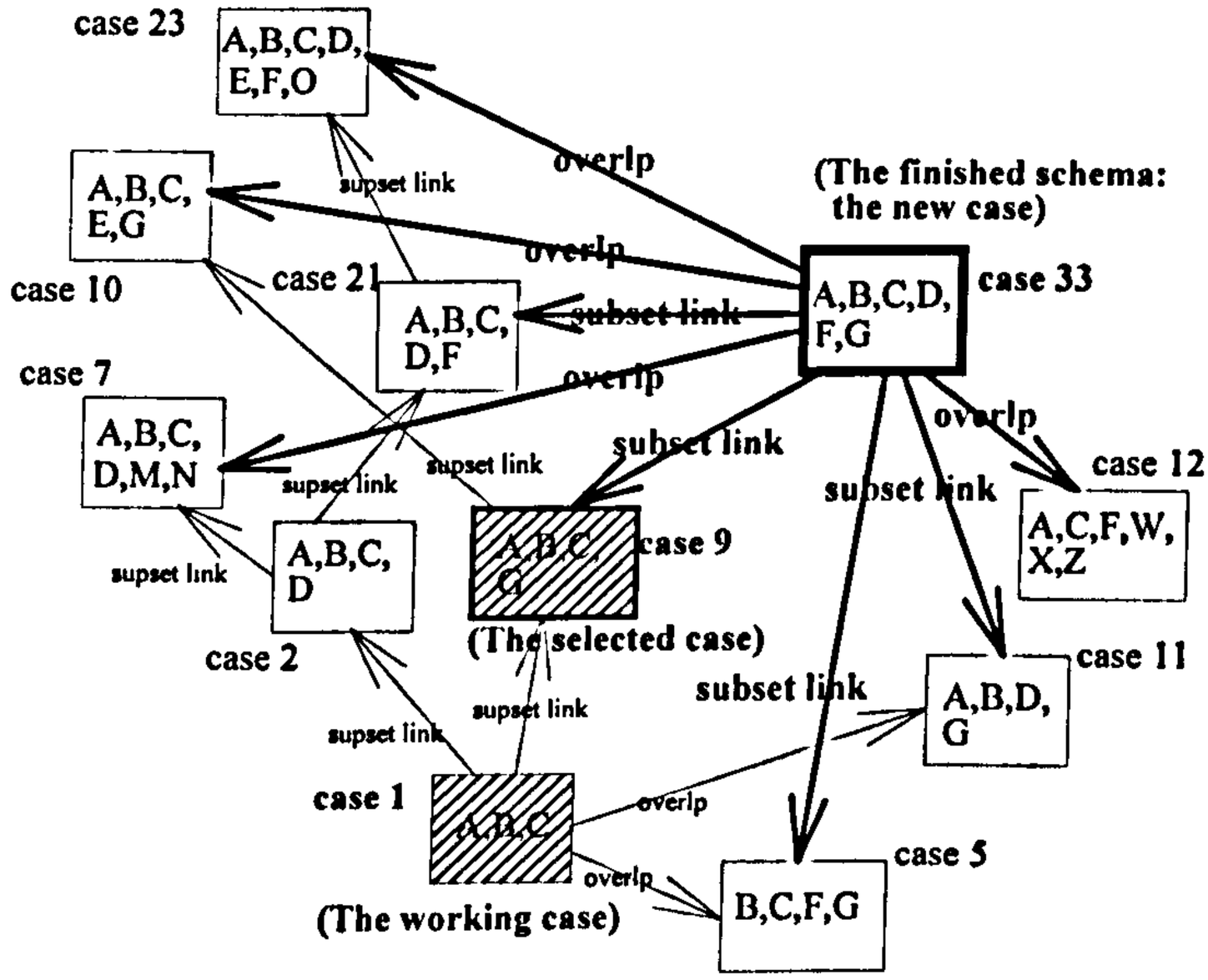
The whole step of the above general rule is carried out systematically by a three-phase process: The first phase is, by means of the supset links of the working case, to find out whether the supset-cases of the working case are of the proper subsets of the finished schema[101]. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of

---

[100] In this rule, the selected case, including some finding element but not the maximum finding element, is of the supremum (the least upper bound) based on the finding element.

[101] Obviously, the subset-cases of the working case are all of the proper subsets of the finished schema.

the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the overlapped cases of the selected case are of the overlapped cases of the new case. An example of this rule is illustrated in Figure B.22.



1. Case 1 is the working case.
2. The selected case, case 9, is of the proper subset of the finished schema and includes the finding element {G,R(G,B)}.
3. The supset-case of the selected case is case 10.
   The overlapped cases of the selected case 9 are case 2, case 5, case 7, case 11, case 12, case 21and case 23.
4. The overlapped cases of the working case 1 are of the proper subsets of the finished schema: case 5 and case 11.
   The supset-cases of the working case 1 are of the proper subsets of the finished schema: case 2 and case 21 (except the selected case 9).
5. Case 2 is of the direct subset-case of case 21.
6. The finished schema, including the new relationship between entity B and entity G, is stored as case 33 in the case base.
   The selected case 9 is a direct subset-case of the new case.
   Case 5 and case 11, which are of the overlapped cases of the selected case 9, are of the direct-subset cases of the new case.
   Case 21, which is of the overlapped case of the selected case, is of the direct-subset case of the new case.
   Case 10, which is of the supset-case of the selected case, is of the overlapped case of the new case.
   Case 7, case 12 and 23, which are of the overlapped cases of the selected case, are of the overlapped cases of the new case.

(For simplification, only some links are shown)

Figure B.22 : An example of G-Rule 7.2.1.4

G-Rule 7.2.1.5 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[102], which is a supset-cases of the working case, is not of the proper subset of the finished schema and together with other selected or ramifying cases cover the maximum finding element[103], but none of them includes the maximum finding element,

then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the selected case depend upon whether these cases are proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case, otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the selected case is of the overlapped case of the new case.

---

[102] See footnote 76.

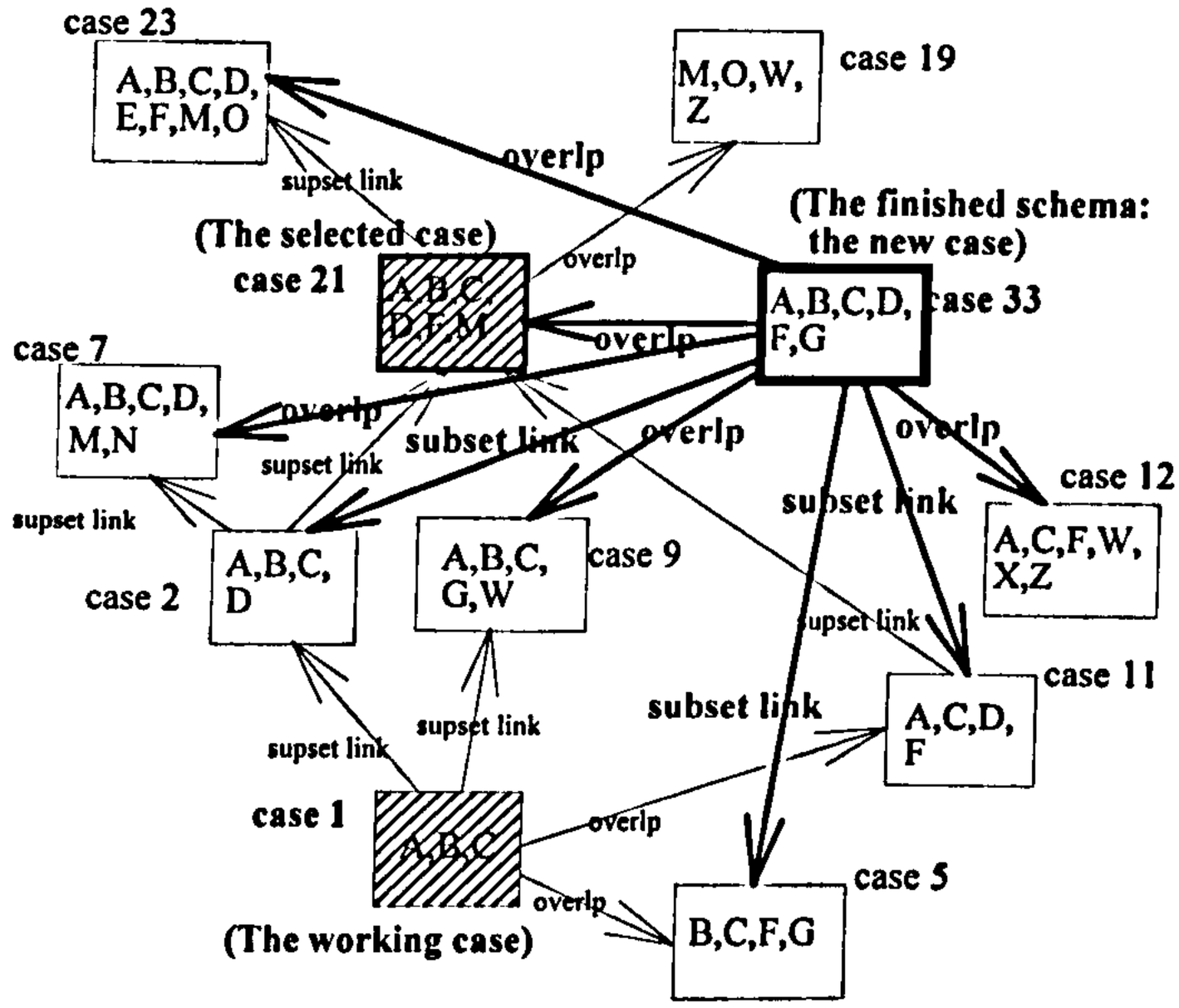
[103] In this rule, the selected case cannot be of the super set of the finished schema. Thus, the selected case must be of the overlapped case of the new case

The process of step (2) in the rule involves travelling down the subset link of the selected case until the first case that is of the proper subset of the finished schema or the working case if the working case is of the proper subset of the finished schema[104], **or** travelling down the subset link of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is **not** of the proper subset of the finished schema[105]. This process will be carried out continuously until all of the subset paths of the selected cases (or the working case) have been checked. Step (3) is carried out in two phases: The first phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of working case are of the proper subset of the finished schema. The second phase is, by using the outcome of the first phase, to find out whether the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is illustrated in Figure B.23.

---

[104] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[105] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 21, is not of the proper subset of the finished schema and includes the finding element {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case 21 is case 23.
   The direct subset-case of the selected case are case 2 and case 11.
   The overlapped cases of the selected case 21 are case 5, case 7, case 9, case 12 and case 19.
4. The first encountered subset-cases of the selected case 21 are of the proper subsets of the finished schema: case 2, case 11.
5. Case 5, case 11 and case 12, which are the overlapped cases of the working case, are the proper subsets of the finished schema.
   Case 19, which is the overlapped case of the selected case, is not relevant to the finished schema.
6. The finished schema, including the new relationship between entity A and entity D, is stored as case 33 in the case base.
   Case 23, which is the direct supset-case of the selected case, is of the overlapped case of the new case.
   Case 2 and case 11, which are of the subset-cases of the selected case, are of the direct subset-case of the new case.
   Case 5, which is of the overlapped case of the selected case, is of the direct subset-case of the new case.
   Case 12, which is of the overlapped case of the selected case, is of the overlapped case of the new case.

(For simplification, only some links are shown)

Figure B.23 : An example of G-Rule 7.2.1.5

B.2.1.4 Circumstance four: the selected case is the overlapped case of the working case

G-Rule 7.2.1.6 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[106], which is the overlapped case of the working case, is of the proper subset of the finished schema and includes the maximum finding element,

then (1) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the super-set cases of the selected case are all of the overlapped case of the new case; (4) the subset-cases of the selected case are all of the subset-cases of the new case; (5) the relation

---

[106] In this rule, the overlapped degree of the selected case of the working case, based on the maximum finding element, is above some threshold value.

types of the overlapped cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the direct subset-case of the new case.

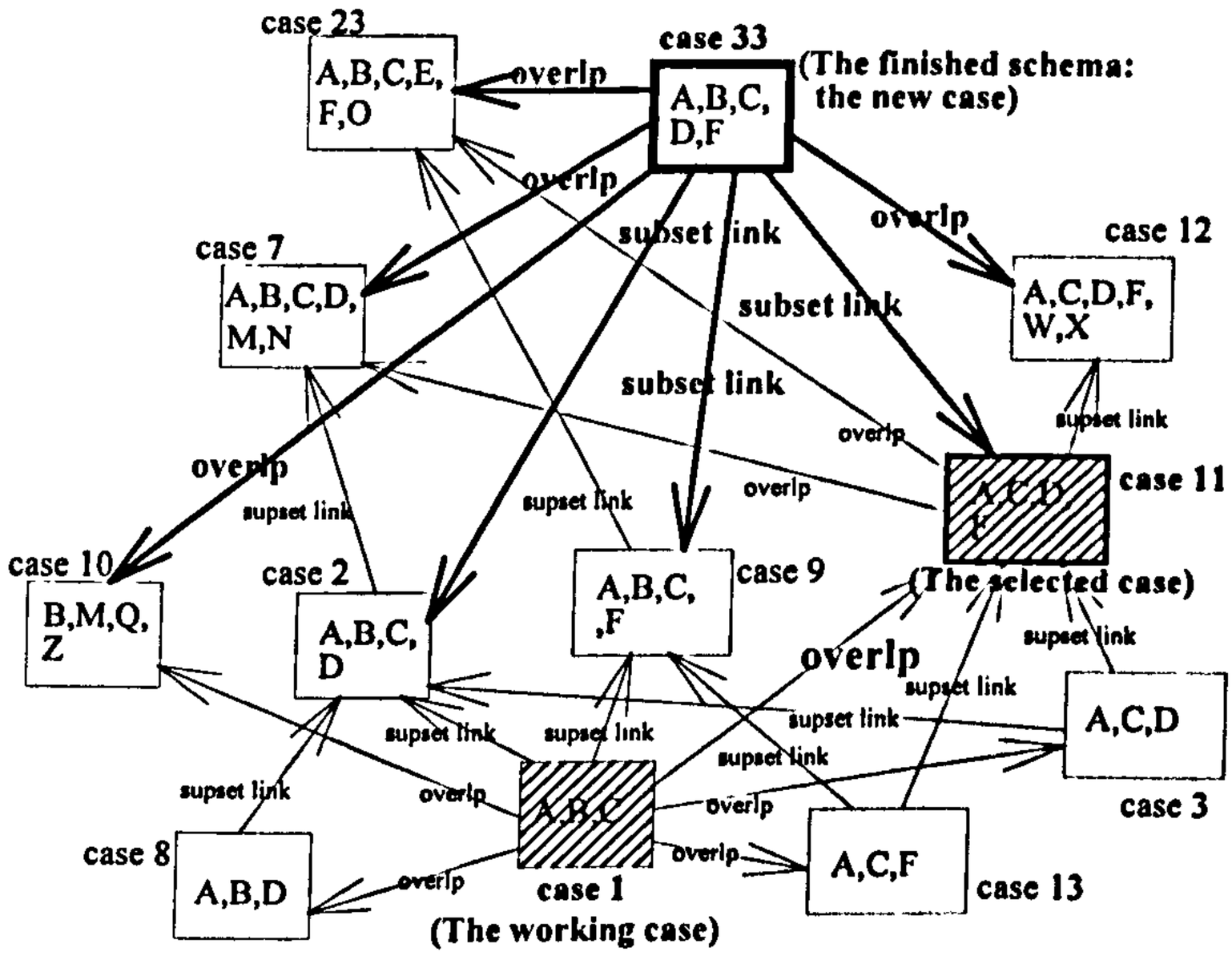
Step (2) to step (5) in the rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[107], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[108]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the overlapped cases of the

---

[107] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new case.

[108] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new case.

selected case are of the overlapped cases of the new case. An example of the rule is illustrated in Figure B.24.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 11, is of the overlapped case of the working case and of the proper subset of the finished schema that includes the maximum finding element {D,F,R(D,A),R(F,C).
3. The supset-case of the selected case is case 12.
   The subset-cases of the selected case are case 3 and case 13.
   The overlapped cases of the selected case are case 2, case 7, case 8, case 9 and case 23.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases (except the selected case) of the working case are case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-cases of the working case are of the proper subsets of the finished schema, and are case 2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 3, case 8, case 13.
6. Case 3 and case 13 are of the subset-cases of the selected case (case 11).
   Case 8 is of the direct subset-cases of the case 2.
7. The finished schema, including the new relationships between entity A and entity D, is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2, case 9 and case 11.
   The overlapped case of the new case are the case 7, case 10, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.24 : An example of G-Rule 7.2.1.6

G-Rule 7.2.1.7 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[109], which is the overlapped case of the working case, is not of the proper subset of the finished schema and includes the maximum finding element, then (1) the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the subset-cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new

---

[109] See footnote 79.

case; (5) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the overlapped case of the new case.

Step (2) to step (6) in the rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[110], or travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is not of the proper subset of the finished schema[111]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the subset-cases and the

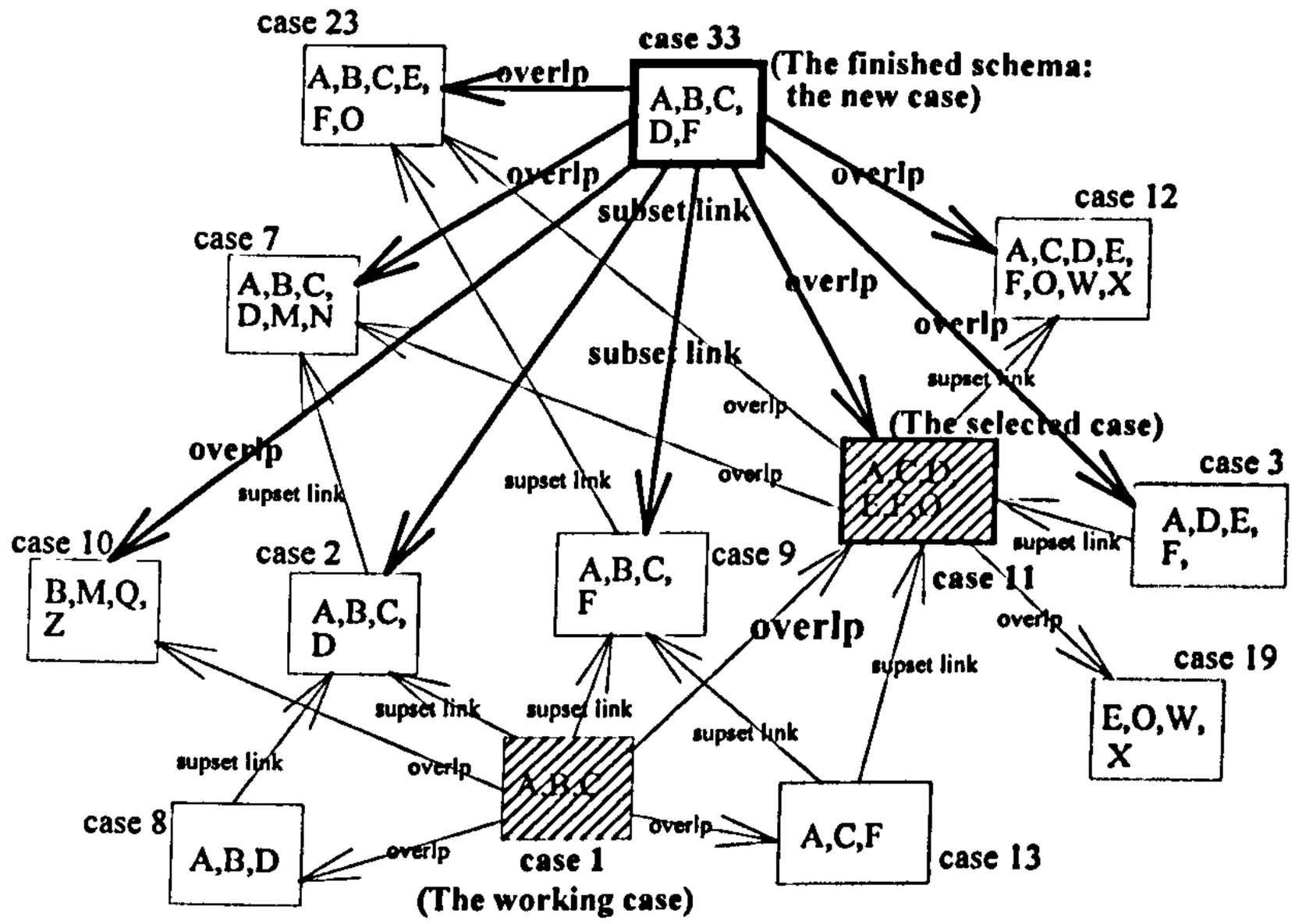
---

[110] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new storing case.

[111] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new storing case.

overlapped cases of the selected case are of the overlapped cases of the new case.

An example of the rule is illustrated in Figure B.25.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 11, is of the overlapped case of the working case and is not of the proper subset of the finished schema that includes the maximum finding element {D,F,R(D,A),R(F,C).
3. The supset-case of the selected case is case 12.
   The subset-cases of the selected case are case 3 and case 13.
   The overlapped cases of the selected case are case 2, case 7, case 8, case 9, case 19 and case 23.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases (except the selected case) of the working case are case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-cases of the working case are of the proper subsets of the finished schema, and are case 2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 8 and case 13.
6. Case13 is of the subset-case of case 9.
   Case 8 is of the direct subset-case of case 2.
   Case 19, which is of the overlapped case of the selected case, is not relevant to the finished case.
7. The finished schema, including the new relationship between entitiy A and entity D, is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2 and case 9.
   The overlapped cases of the new case are case 3, case 7, case 10, case 11, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.25 : An example of G-Rule 7.2.1.7

G-Rule 7.2.1.8 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case, which is the overlapped case of the working case, is of the proper subset of the finished schema and, together with other selected or ramifying selected cases, covers the maximum finding element, but none of them includes the maximum finding element, then (1) the relation types of the supset-cases and the subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the supset-cases of the selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the overlapped cases of the selected

case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (5) the selected case, if none of the subset-cases of the new case is of the supset-case of the selected case, is of the direct subset-case of the new case.

Step (1) to step (5) in the above general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[112], **or** travelling down the subset links of the working case until the first case that is of the proper subset of the finish schema or the end case of the path if the working case is not of the proper subset of the finished schema[113]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the supset-cases and the

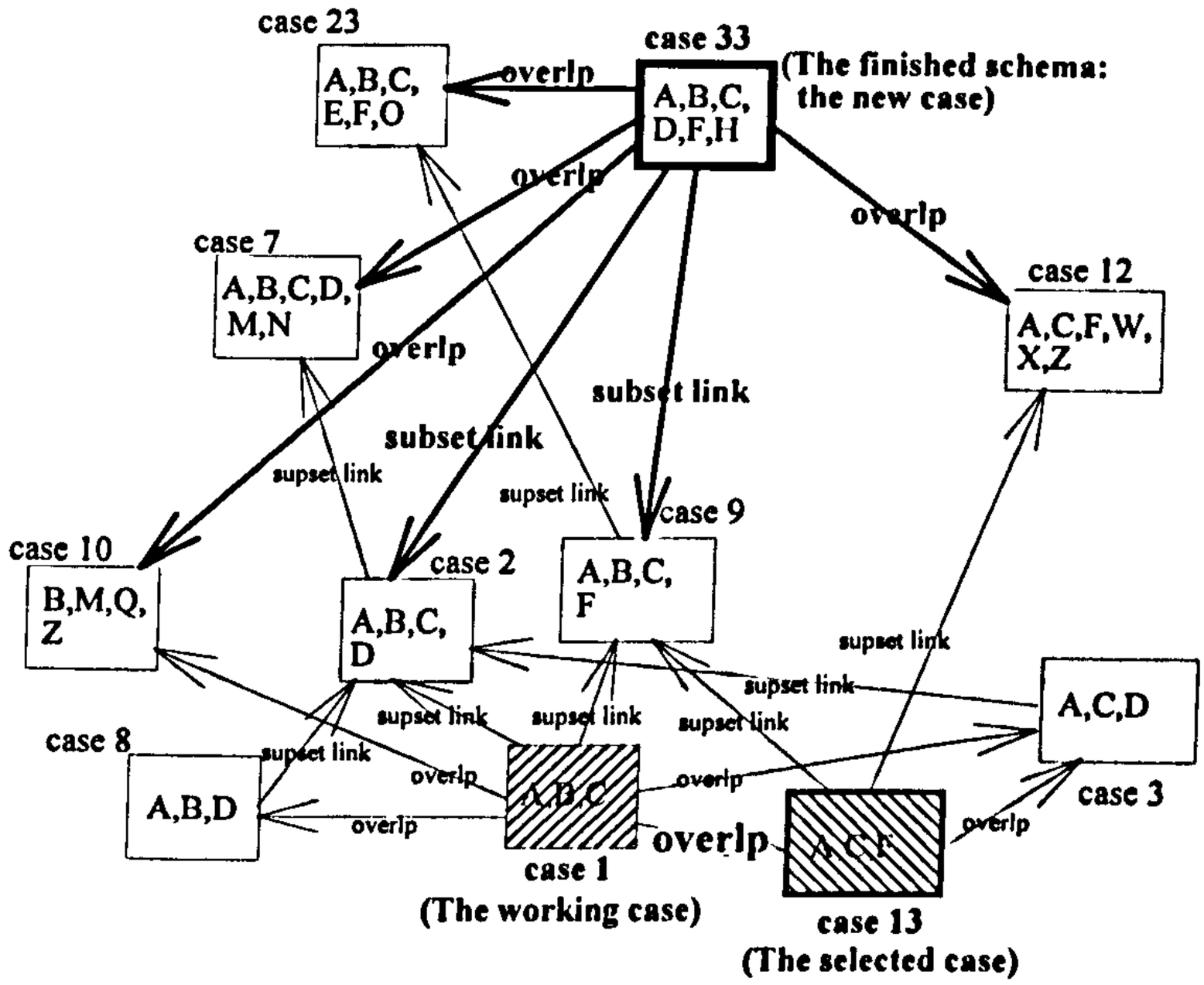
---

[112] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new storing case.

[113] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new storing case.

overlapped cases of the selected case are of the overlapped cases of the new case.

An example of the rule is illustrated in Figure B.26.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 13, is of the overlapped case of the working case, being of the proper subsets of the finished schema and including some finding elements {F,R(F,C)}.
3. The supset-cases of the selected case 13 are case 9, case 12 and case 23.
   The overlapped cases of the selected case 13 are case 1, case 2, case 3, case 7 and case 8.
4. The supset-cases of the working case are case 2, case 7, case 9 and case 23.
   The overlapped cases of the working case are case 3, case 8, case 10, case 12 and case 13.
5. By means of the supset links of the working case, the supset-cases of the working case are of the proper subset of the finished schema, and are case 2 and case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 3, case 8, case 13.
6. Case 3 and case 8 are the subset-cases of case 2.
   Case 13 is the subset-case of case 9.
7. The finished schema, including the new entity H for the case base (i.e. the corresponding concept h for the concept base), is stored as case 33 in the case base.
   The direct subset-cases of the new case are case 2 and case 9.
   The overlapped cases of the new case are case 7, case 10, case 12 and case 23.

(For simplification, only some links are shown)

Figure B.26 : An example of G-Rule 7.2.1.8

G-Rule 7.2.1.9 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the selected case[114], which is the overlapped case of the working case, is not of the proper subset of the finished schema and, together with other selected or ramifying selected cases, covers the maximum finding element, but none of them includes the maximum finding element,

then (1) the relation types of the supset-cases of the selected case are all of the overlapped case of the new case; (2) the relation types of the supset-cases and subset-cases of the working case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the working case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the relation types of the subset-cases of the selected case depend upon whether

---

[114] See footnote 79.

these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (5) the relation types of the overlapped cases of the selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (6) the selected case is of the overlapped case of the new case.

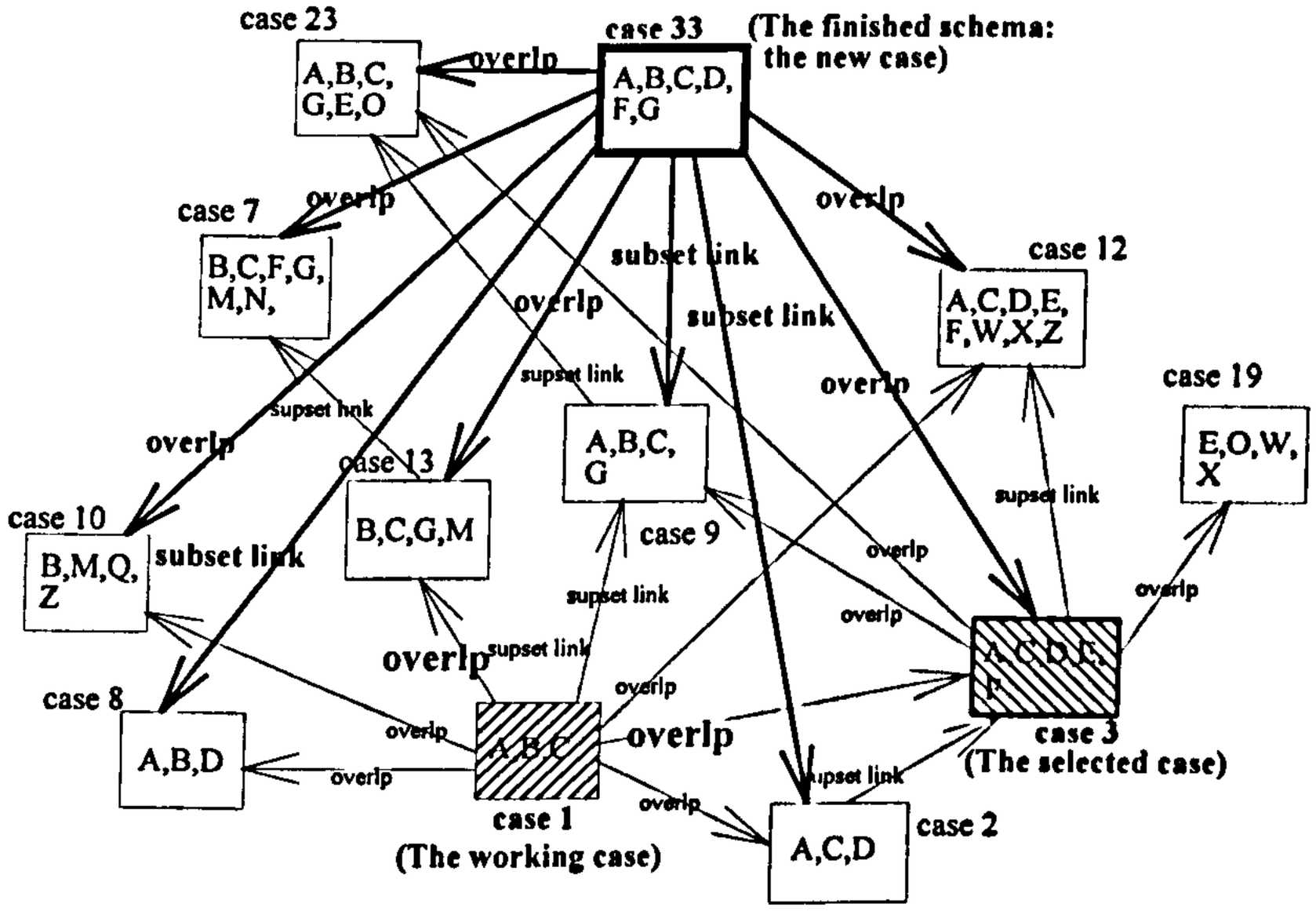
Step (2) to step (6) in this general rule are carried out systematically by a three-phase process: The first phase involves travelling up the supset links of the working case until the first case that is not of the proper subset of the finished schema if the working case is of the proper subset of the finished schema[115], **or** travelling down the subset links of the working case until the first case that is of the proper subset of the finished schema or the end case of the path if the working case is **not** of the proper subset of the finished schema[116]. This process will be carried out continuously until all of the supset or subset paths of the working case have been checked. The second phase is, by means of the overlapped links of the working case, to find out whether the overlapped cases of the working case are of

---

[115] In this situation, the subset-cases of the working case are obviously all of the subset-cases of the new storing case.

[116] In this situation, the supset-cases of the working case are obviously all of the overlapped cases of the new storing case.

the proper subsets of the finished schema. The third phase, by using the outcome of the first and second phases, aims to find out whether the subset-cases and the overlapped cases of the selected case are of the overlapped cases of the new case. An example of the rule is illustrated in Figure B.27.



1. Case 1 is the working case and is of the proper subset of the finished schema.
2. The selected case, case 3, is of the overlapped case of the working case and is not of the proper subsets of the finished schema that includes the finding element {D,F,R(D,A),R(F,C)}.
3. The supset-case of the selected case 3 is case 12.
   The subset-case of the selected case 3 is case 2.
   The overlapped cases of the selected case 3 are case 7, case 8, case 9, case 13, case 19 and case 23.
4. The supset-cases of the working case are case 9 and case 23.
   The overlapped cases (except the selected cases) of the working case are case 2, case 7, case 8, case 10 and case 12.
5. By means of the supset links of the working case, the supset-case of the working case is of the proper subset of the finished schema: case 9.
   By means of the overlapped links of the working case, the overlapped cases of the working case are of the proper subsets of the finished schema, and are case 2 and case 8.
6. Case 19, which is of the overlapped case of the selected case, is not relevant to the finished schema.
   The direct subset-cases of the new storing case are case 2, case 8 and case 9.
7. The finished schema, including the new relationship between entity A and entity D, is stored as case 33 in the case base.
   The overlapped cases of the new storing case are case 3, case 7, case 10, case 12, case 13 and case 23.

(For simplification, only some links are shown)

Figure B.27 : An example of G-Rule 7.2.1.9

B.2.2 The Ramifying Selected Case That is not Relevant to the Working Case

There are two circumstances in this situation, depending on the type of cases which the ramifying selected cases are from. The first circumstance is where the ramifying selected cases are branched from some selected case. The second circumstance is where the ramifying selected cases are branched from some ramifying selected case.

Thus, the ramifying selected case in this situation must be *either* of the overlapped case(s) of the selected cases[117] that are the supset-cases or the overlapped cases of the working case *or* of the overlapped case(s) of the other ramifying cases[118] that are the overlapped case(s) of the selected case. In order to locate the relevant cases and to create the relation types of these relevant cases for storing a new case, the primitive general rules illustrated below are mixed with the primitive rules illustrated in situation (I) to handle the number of combinations in which the elements of the finished schema might be from the old cases which are in the situation (I) or/and situation (II).

---

[117] If a ramifying selected case is a supset-case of the selected cases, the ramifying selected case will be viewed as the selected case in the storing stage.

[118] If the ramifying selected case is a supset-case of the other ramifying selected case, the super ramifying selected case will be viewed as the ramifying selected case in the storing stage.

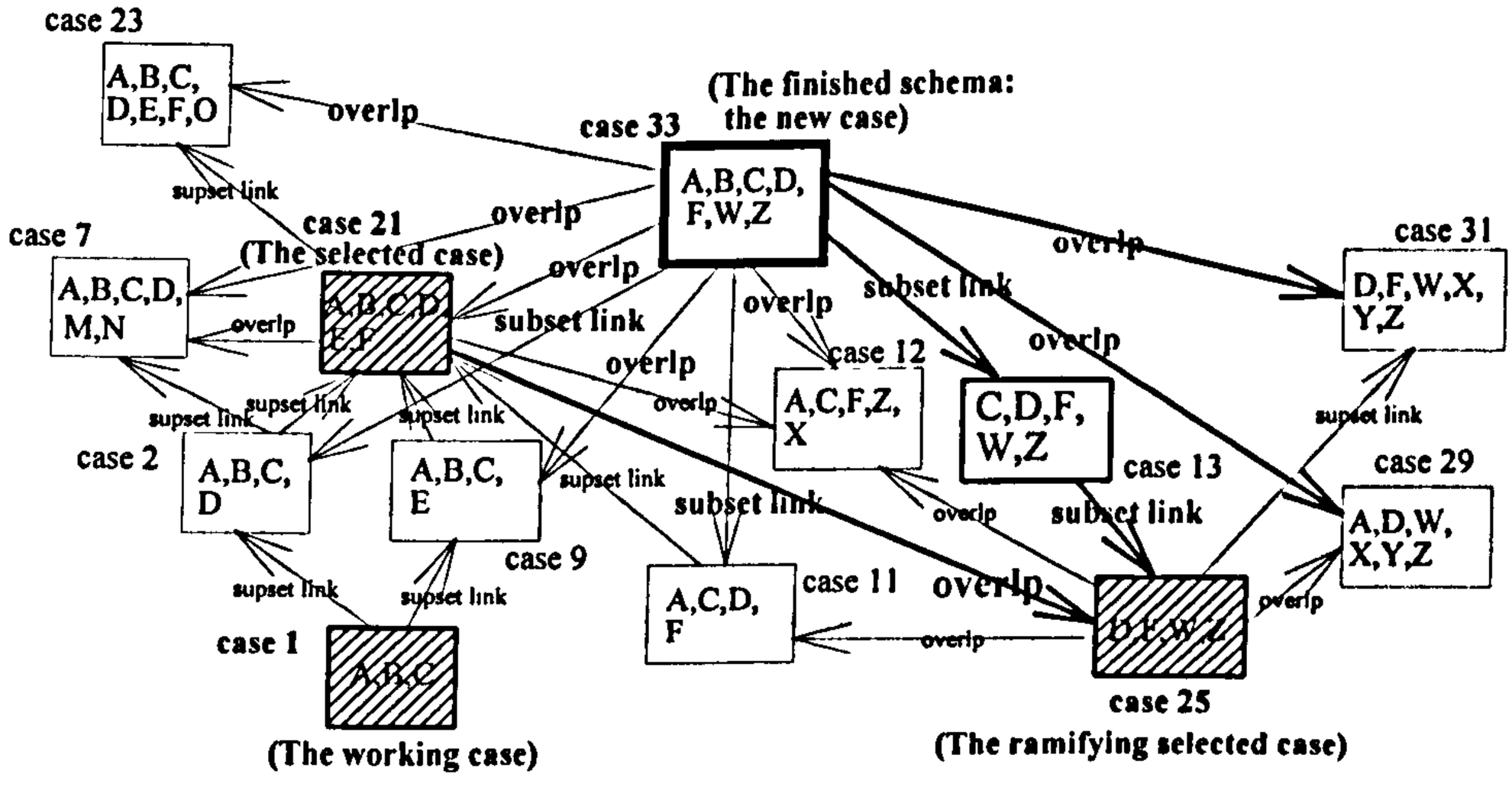
B.2.2.1 Circumstance one: the ramifying selected case is from the selected case

G-Rule 7.2.2.1 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the supset-case of the working case, and is either of the subset-case or the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the ramifying selected case are all of the subset-case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case, if none of its supset-

cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the general rule is shown in Figure B.28. The whole process is carried out systematically by two phases: The first phase is, by means of G-Rule 7.2.1.4 (the selected case, which is the supset-case of the working case, is of the subset-case of the new case) or G-Rule 7.2.1.5 (the selected case, which is the supset-case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and overlapped cases of the ramifying selected case for the new case.



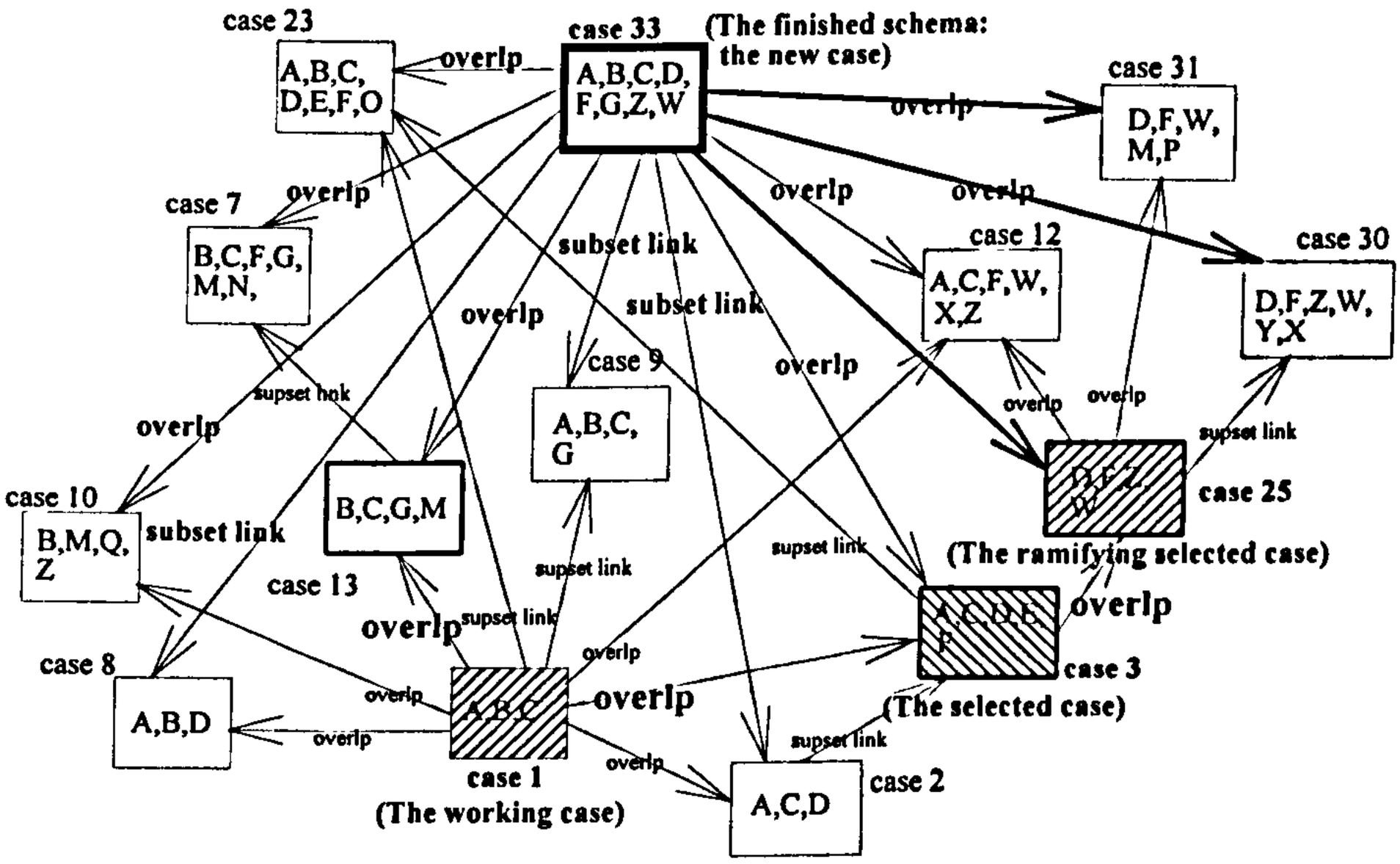
(For simplification, only some links are shown)

Figure B.28 : An example of G-Rule 7.2.2.1

G-rule 7.2.2.2 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the overlapped case of the working case, and is either of the subset-case or the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the ramifying selected case are all of the subset-case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case, if none of its supset-cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the general rule is shown in Figure B.29. The whole process is carried out systematically by two phases: The first phase is, by means of G-Rule 7.2.1.8 (the selected case, which is the overlapped case of the working case, is of the subset-case of the new case) or G-Rule 7.2.1.9 (the selected case, which is the overlapped case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.29 : An example of G-Rule 7.2.2.2

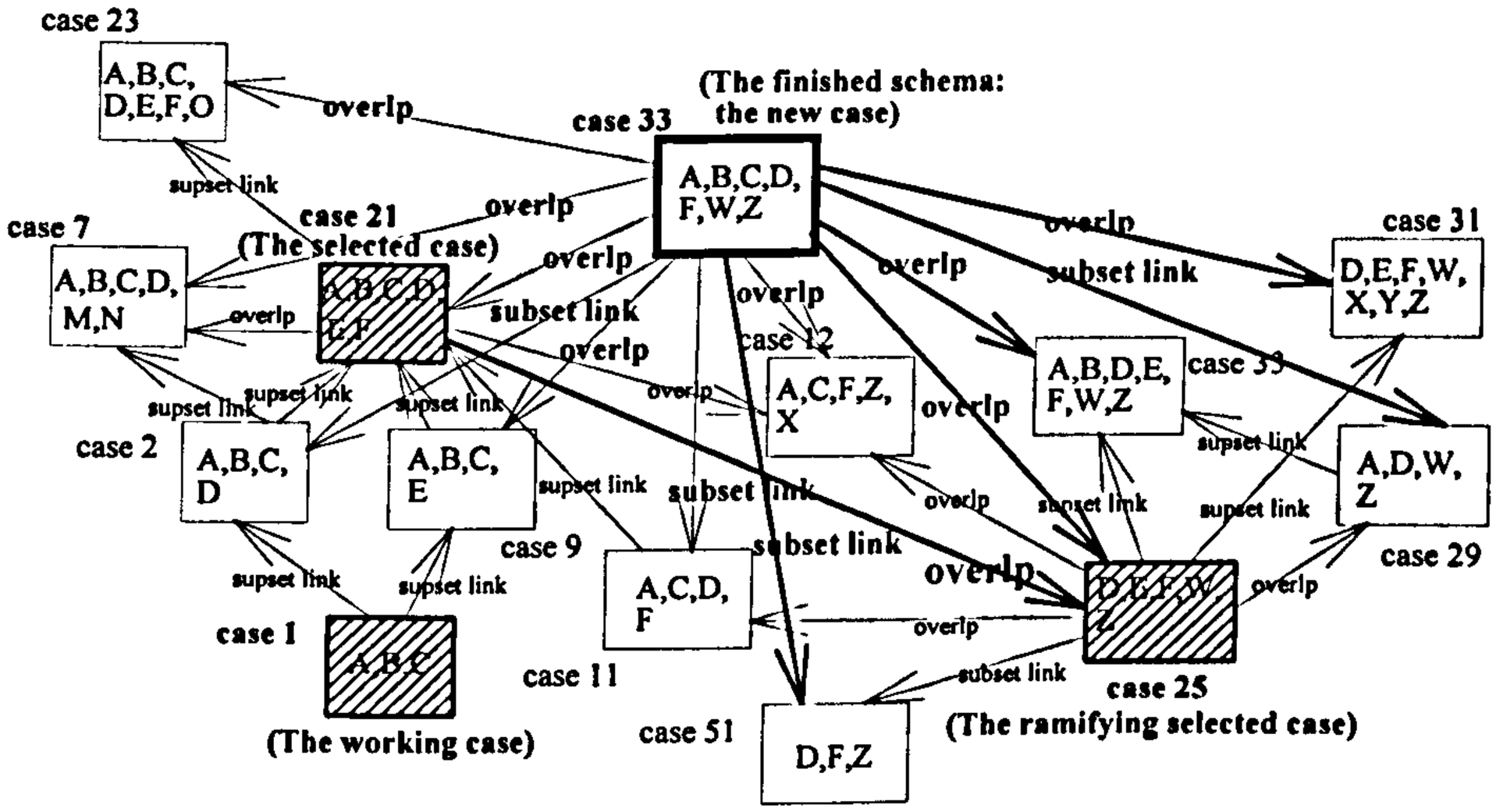
G-Rule 7.2.2.3 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the supset-case of the

working case, and is either of the subset-case or the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case is of the overlapped case of the new case.

An example of the rule is shown in Figure B.30. The whole process is carried out systematically by two phases: The first phase is, by means of G-Rule 7.2.1.4 (the selected case, which is the supset-case of the working case, is of the subset-case of the new case) or G-Rule 7.2.1.5 (the selected case, which is the supset-case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the

outcome of the first phase, to find the relation types of the subset-cases and the overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

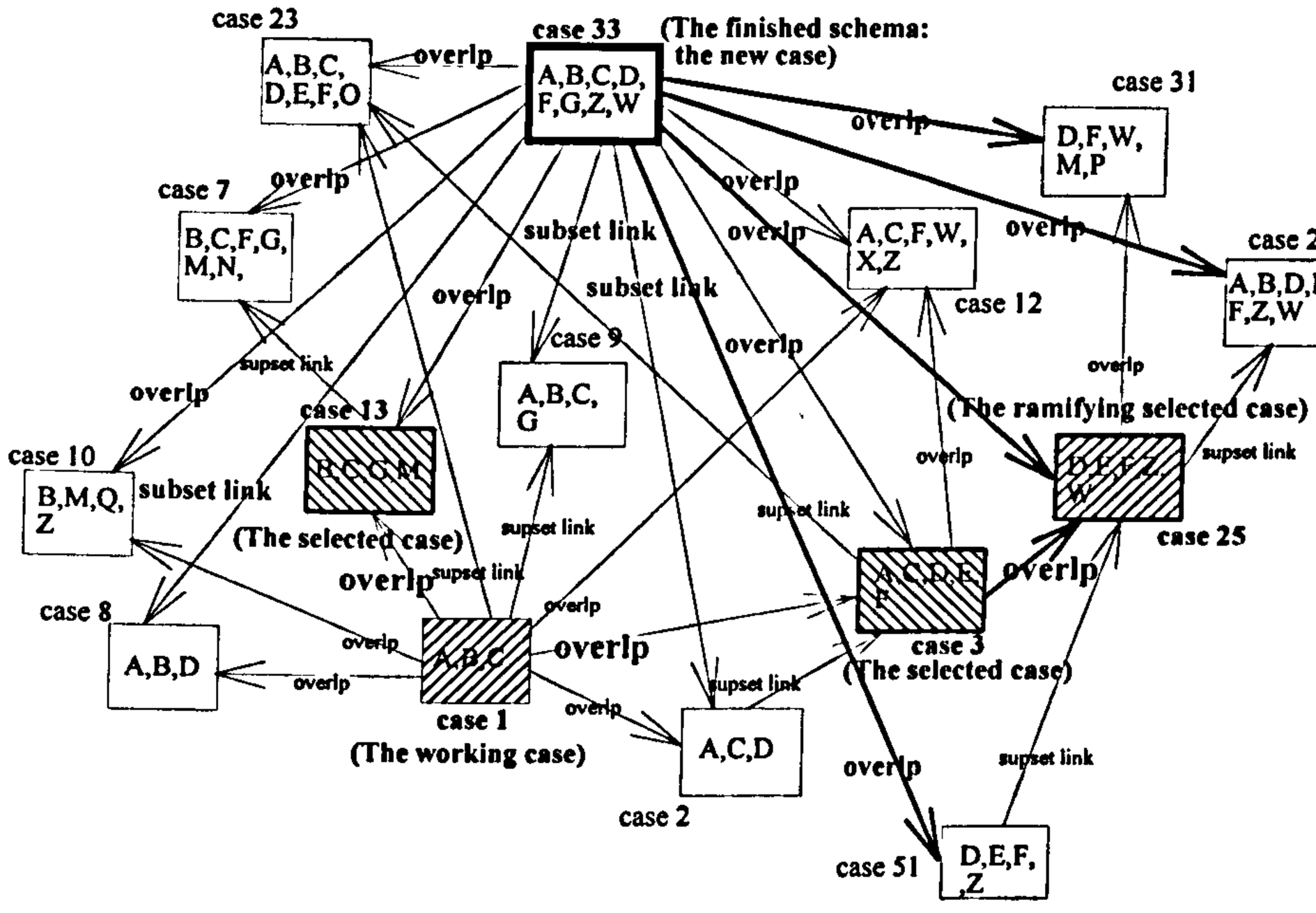
Figure B.30 : An example of G-Rule 7.2.2.3

G-rule 7.2.2.4 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the ramifying selected case, which is the overlapped case of the selected case, which is of the overlapped case of the working case, and is either of the subset-case or the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the ramifying case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the ramifying selected case depend upon whether these cases are relevant to the finished

schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the ramifying selected case is of the overlapped case of the new case.

An example of the rule is shown in Figure B.31. The whole process is carried out systematically by two phases: The first phase is, by means of G-Rule 7.2.1.8 (the selected case, which is the overlapped case of the working case, is of the subset-case of the new case) or G-Rule 7.2.1.9 (the selected case, which is the overlapped case of the working case, is of the overlapped case of the new case), to check the relation types of the relevant cases of the selected case. The second phase is, by using the outcome of the first phase, to find the relation types of the subset-cases and the overlapped cases of the ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.31 : An example of G-Rule 7.2.2.4

B.2.2.2 Circumstance two: the ramifying selected case is from another ramifying selected case

In this circumstance, the ramifying selected cases are classified into two types. The first type is called the *original ramifying selected case*, abbreviated as *O-ramifying selected case*. The second type is called the *destination ramifying selected case*, abbreviated as *D-ramifying selected case*[119].

G-Rule 7.2.2.5 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the D-

---

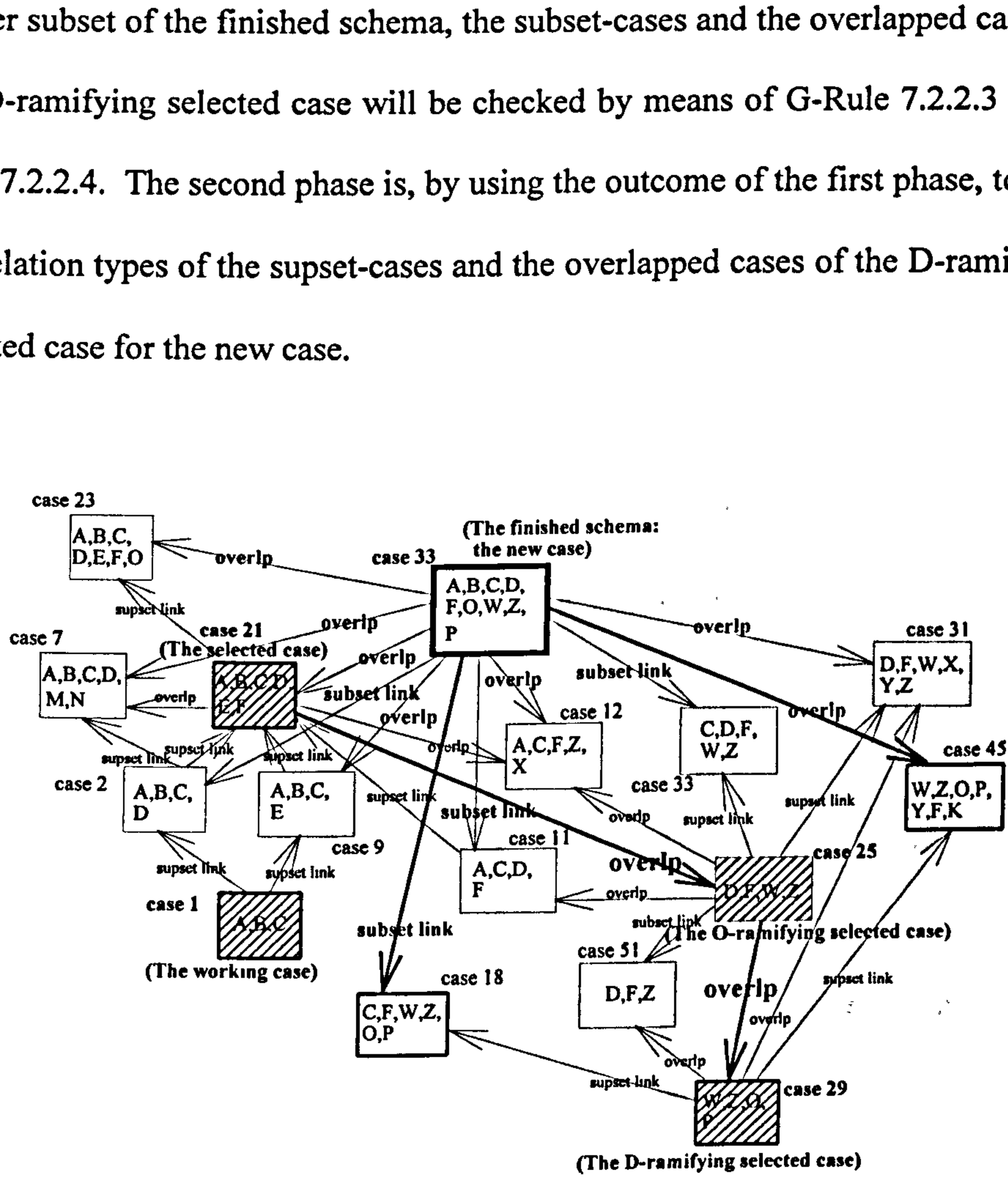
[119] Basically, the O-ramifying selected case can be viewed as a working case of the D-ramifying selected case.

ramifying selected case, which is the overlapped case of the O-ramifying selected case, which is either of the subset-case or the overlapped case of the new case, is of the proper subset of the finished schema,

then (1) the supset-cases of the D-ramifying selected case depend upon whether these supset-cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (2) the subset-cases of the D-ramifying selected case are of the subset-case of the new case; (3) the relation types of the overlapped cases of the D-ramifying selected case depend upon whether these cases are of the proper subsets of the finished schema. If the elements of the case are included in the new case, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the D-ramifying selected case, if none of its supset-cases is of the subset-case of the finished schema, is of the direct subset-case of the new case.

An example of the rule is shown in Figure B.32. The whole process is carried out systematically in two phases: The first phase is, according to the relation types of the O-ramifying selected case (i.e. the O-ramifying selected case is a subset-case or an overlapped case of the finished schema), to check the relation types of the

relevant cases of the O-ramifying selected case. For example, if the O-ramifying selected case is of the proper subset of the finished schema, the supset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.2.2.1 or G-Rule 7.2.2.2. If the O-ramifying selected case is not of the proper subset of the finished schema, the subset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.2.2.3 or G-Rule 7.2.2.4. The second phase is, by using the outcome of the first phase, to find the relation types of the supset-cases and the overlapped cases of the D-ramifying selected case for the new case.



(For simplification, only some links are shown)

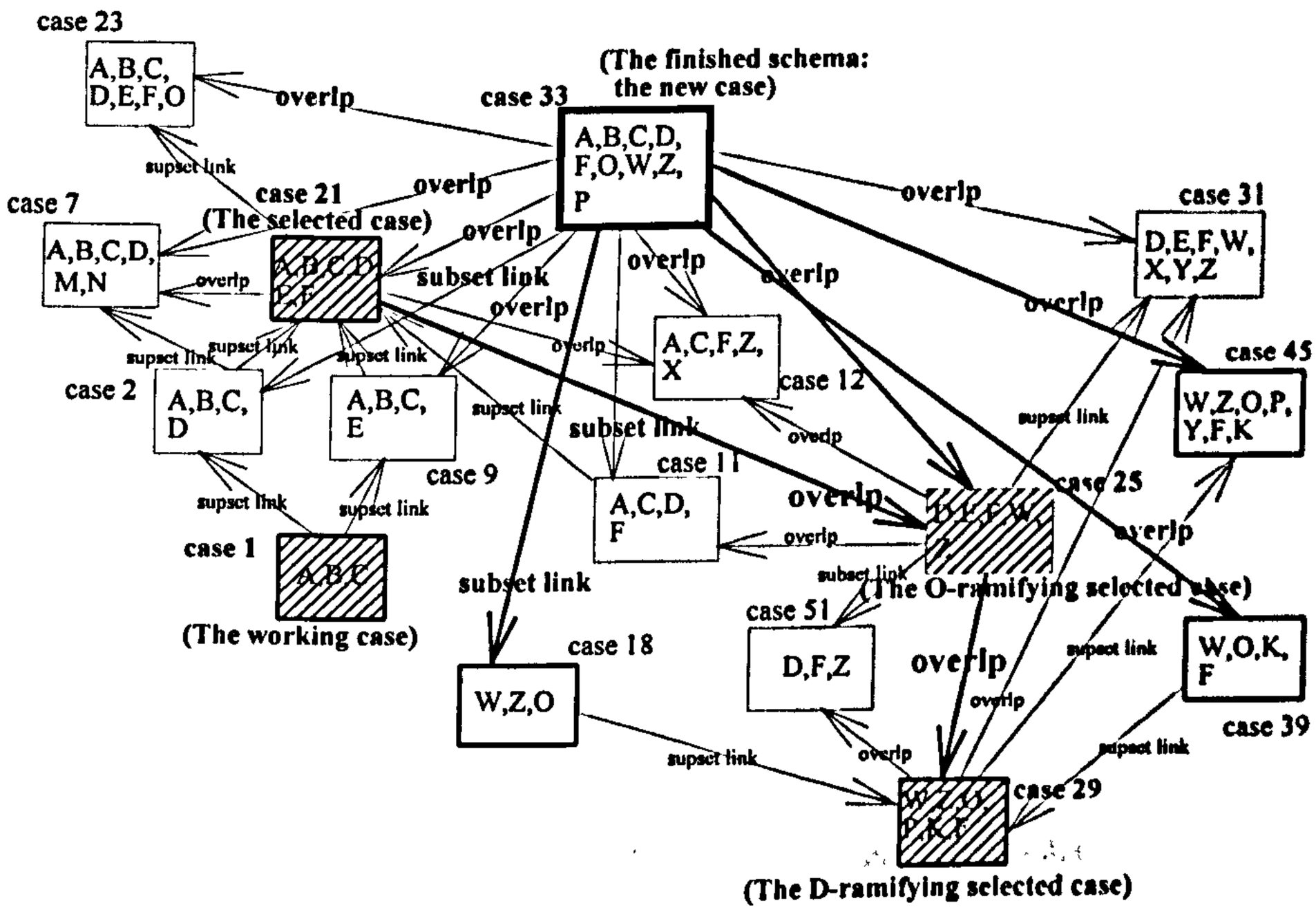
Figure B.32 : An example of G-Rule 7.2.2.5

G-Rule 7.2.2.6 : If (1) the finished schema includes the new property of the relationship between related entities, and (2) the D-ramifying selected case, which is the overlapped case of the O-ramifying selected case, which is either of the subset-case or the overlapped case of the new case, is not of the proper subset of the finished schema,

then (1) the supset-cases of the D-ramifying case are all of the overlapped case of the new case; (2) the relation types of the subset-cases of the D-ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (3) the relation types of the overlapped cases of the D-ramifying selected case depend upon whether these cases are relevant to the finished schema. If so, and if of the proper subset of the finished schema, it is a subset-case of the new case; otherwise it is the overlapped case of the new case; (4) the D-ramifying selected case is of the overlapped case of the new case.

An example of the rule is shown in Figure B.33. The whole process is carried out systematically in two phases: The first phase is, according to the relation types of the O-ramifying selected case (i.e. the O-ramifying selected case is a subset-case

or an overlapped case of the finished schema), to check the relation types of the relevant cases of the O-ramifying selected case. For example, if the O-ramifying selected case is of the proper subset of the finished schema, the supset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.2.2.1 or G-Rule 7.2.2.2. If the O-ramifying selected case is not of the proper subset of the finished schema, the subset-cases and the overlapped cases of the O-ramifying selected case will be checked by means of G-Rule 7.2.2.3 or G-Rule 7.2.2.4. The second phase is, by using the outcome of the first phase, to find the relation types of the subset-cases and the overlapped cases of the D-ramifying selected case for the new case.



(For simplification, only some links are shown)

Figure B.33 : An example of G-Rule 7.2.2.6

B.3 The Finished Schema Involves a New Property of the Relationship Between Known Related Entities (the first situation of the second aspect of influencing the knowledge accumulation)

G-Rule 7.3 : If the difference between the finished schema and the selected stored case (only this selected case in the whole design session) is only in the property of a known relationship between known entities,

then the new property of a known relationship of the relevant entities in the selected case is annotated.

An example of the rule is depicted in Figure B.34.

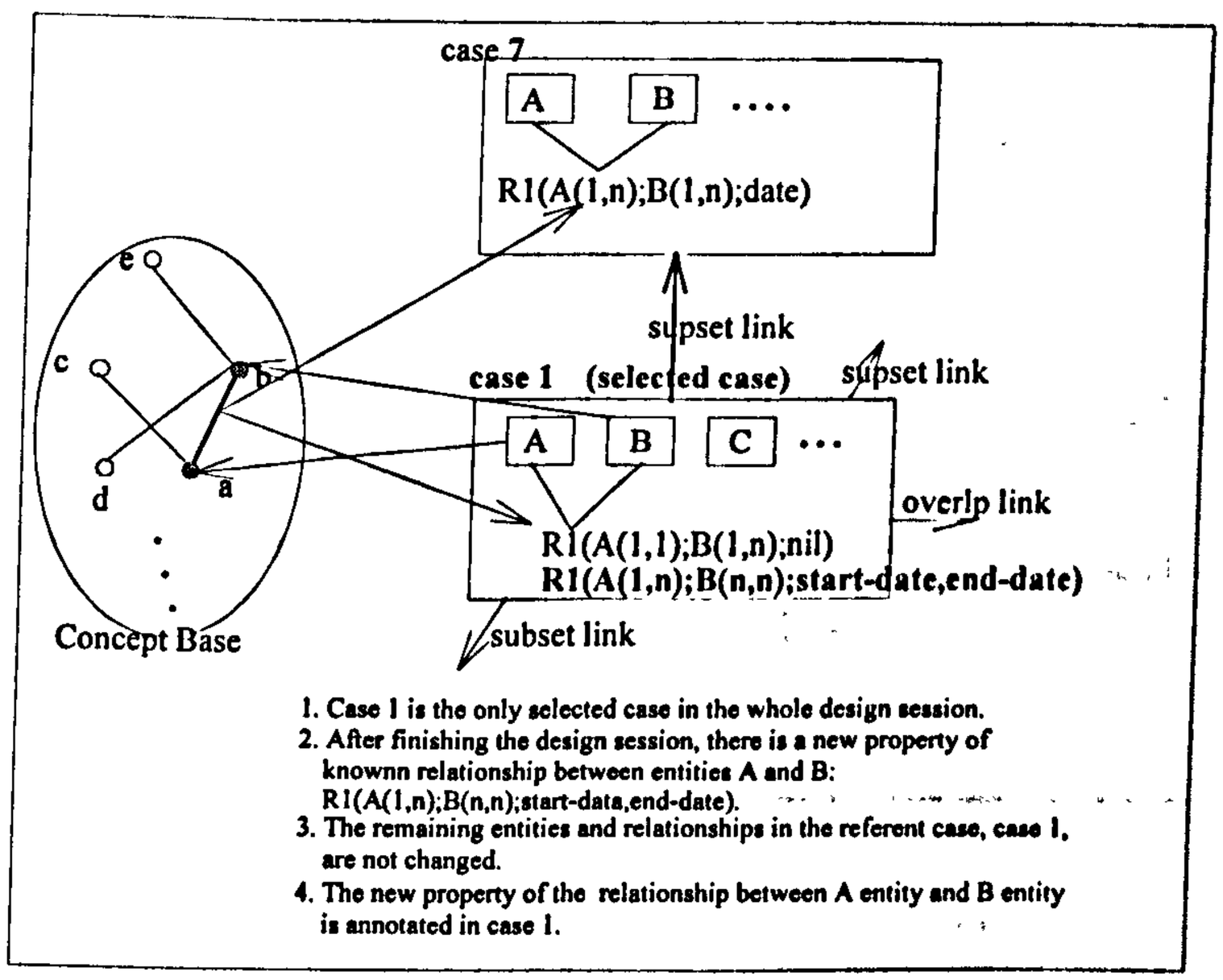


Figure B.34 : An example of G-Rule 7.3

If this annotated relationship of the entities is applied to the new case in the later design session, the annotated relationship of the entities will be deleted.

B.4 The Finished Schema Only Involves a New Exemplar of the Known Concept in the Concept Base

G-Rule 7.4 : If there is a new exemplar of the known concept, then a non-case entity and an exemplar link of the corresponding known concept are created.

An example of the rule is depicted in Figure B.35.

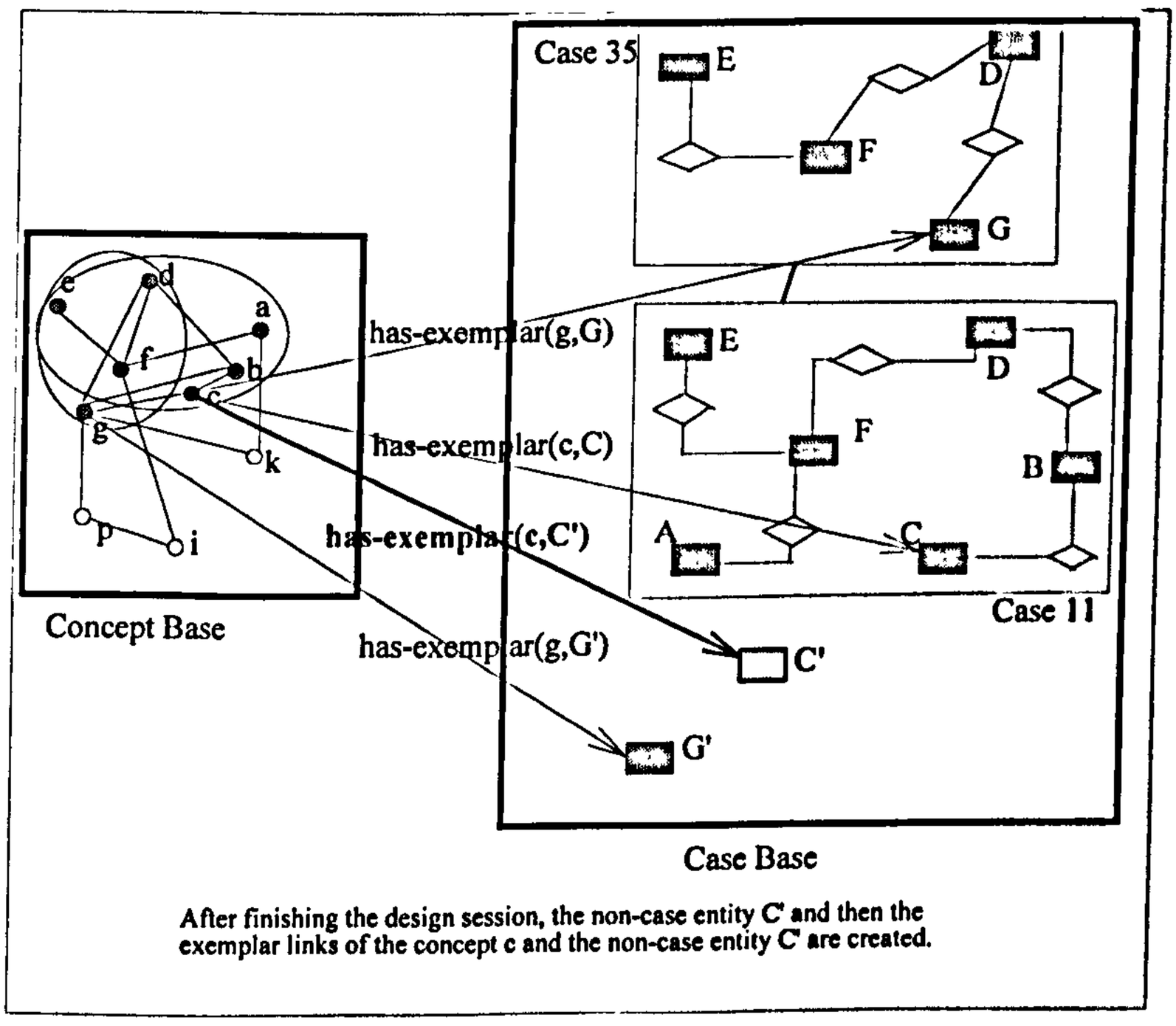


Figure B.35 : An example of G-Rule 7.4

The rule is applied only if the attributes of the new exemplar (entity) are not included in the stored cases. If the content of the non-case entity is copied as the content of an entity in the new case in the later design session, the non-case entity and the exemplar link of the corresponding concept will be deleted.