THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *20/03/2015* **par :**

EZGI IRAZ SU

Extensions of Equilibrium Logic by Modal Concepts

JURY

PHILIPPE BALBIANI	Directeur de Recherche - CNRS
LUIS FARINAS DEL CERRO	Directeur de Recherche - CNRS
OLIVIER GASQUET	Professeur - Université Paul Sabatier
ANDREAS HERZIG	Directeur de Recherche - CNRS
DAVID PEARCE	Professeur - Universidad Politécnica de Madrid
TORSTEN SCHAUB	Professeur - Universität Potsdam
AGUSTÍN VALVERDE	Professeur - Universidad de Malaga

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse

Directeur(s) de Thèse :
Luis FARINAS del CERRO, *Andreas HERZIG* et *David PEARCE*

Rapporteurs :
Torsten SCHAUB et *Agustín VALVERDE RAMOS*

Bir tanecik teyzem Sacit'ime
ve canım anneme...

Abstract

Here-and-there (**HT**) logic is a three-valued monotonic logic which is intermediate between classical logic and intuitionistic logic. Equilibrium logic is a nonmonotonic formalism whose semantics is given through a minimisation criterion over HT models. It is closely aligned with answer set programming (**ASP**), which is a relatively new paradigm for declarative programming. To spell it out, equilibrium logic provides a logical foundation for **ASP**: it captures the answer set semantics of logic programs and extends the syntax of answer set programs to more general propositional theories, i.e., finite sets of propositional formulas. This dissertation addresses modal logics underlying equilibrium logic as well as its modal extensions. It allows us to provide a comprehensive framework for **ASP** and to reexamine its logical foundations.

In this respect, we first introduce a monotonic modal logic called **MEM** that is powerful enough to characterise the existence of an equilibrium model as well as the consequence relation in equilibrium models. The logic **MEM** thus captures the minimisation attitude that is central in the definition of equilibrium models.

Then we introduce a dynamic extension of equilibrium logic. We first extend the language of **HT** logic by two kinds of atomic programs, allowing to update the truth value of a propositional variable here or there, if possible. These atomic programs are then combined by the usual dynamic logic connectives. The resulting formalism is called dynamic here-and-there logic (**D-HT**), and it allows for atomic change of equilibrium models. Moreover, we relate **D-HT** to dynamic logic of propositional assignments (**DL-PA**): propositional assignments set the truth values of propositional variables to either true or false and update the current model in the style of dynamic epistemic logics. Eventually, **DL-PA** constitutes an alternative monotonic modal logic underlying equilibrium logic.

In the beginning of the 90s, Gelfond has introduced *epistemic specifications* (**E-S**) as an extension of disjunctive logic programming by epistemic notions. The underlying idea of **E-S** is to correctly reason about incomplete information, especially in situations when there are multiple answer sets. Related to this aim, he has proposed the *world view* semantics because the previous *answer set* semantics was not powerful enough to deal with commonsense reasoning. We here add epistemic operators to the original language of **HT** logic and define an epistemic version of equilibrium logic. This provides a new semantics not only for Gelfond's epistemic specifications, but also for more general nested epistemic logic programs. Finally, we compare our approach with the already existing semantics, and also provide a strong equivalence result for EHT theories. This paves the way from **E-S** to epistemic **ASP**, and can be regarded as a nice starting point for further frameworks of extensions of **ASP**.

Résumé

La logique *Here-and-there* (**HT**) est une logique monotone à trois valeurs, intermédiaire entre les logiques intuitionniste et classique. La logique de l'*équilibre* est un formalisme non-monotone dont la sémantique est donnée par un critère de minimalisation sur les modèles de la logique **HT**. Ce formalisme est fortement lié à la programmation orientée ensemble réponse (**ASP**), un paradigme relativement nouveau de programmation déclarative. La logique de l'équilibre constitue la base logique de l'**ASP**: elle reproduit la sémantique par ensemble réponse des programmes logiques et étend la syntaxe de l'**ASP** à des théories propositionnelles plus générales, i.e., des ensembles finis de formules propositionnelles. Cette thèse traite aussi bien des logiques modales sous-jacentes à la logique de l'équilibre que de ses extensions modales. Ceci nous permet de produire un cadre complet pour l'**ASP** et d'examiner de nouveau la base logique de l'**ASP**.

A cet égard, nous présentons d'abord une logique modale monotone appelée **MEM** et capable de caractériser aussi bien l'existence d'un modèle de la logique de l'équilibre que la relation de conséquence dans ces modèles. La logique **MEM** reproduit donc la propriété de minimalisation qui est essentielle dans la définition des modèles de la logique de l'équilibre.

Nous définissons ensuite une extension dynamique de la logique de l'équilibre. Pour ce faire, nous étendons le langage de la logique **HT** par deux ensembles de programmes atomiques qui permettent de mettre à jour, si possible, les valeurs de vérité des variables propositionnelles. Ces programmes atomiques sont ensuite combinés au moyen des connecteurs habituels de la logique dynamique. Le formalisme résultant est appelé logique *Here-and-there dynamique* (**D-HT**) et permet la mise-à-jour des modèles de la logique de l'*équilibre*. Par ailleurs, nous établissons un lien entre la logique **D-HT** et la *logique dynamique des affectations propositionnelles* (**DL-PA**): les affectations propositionnelles mettent à vrai ou à faux les valeurs de vérité des variables propositionnelles et transforment le modèle courant comme en logique dynamique propositionnelle. En conséquence, **DL-PA** constitue également une logique modale sous-jacente à la logique de l'équilibre.

Au début des années 1990, Gelfond avait défini les spécifications épistémiques (**E-S**) comme une extension de la programmation logique disjonctive par des notions épistémiques. L'idée de base des **E-S** est de raisonner correctement à propos d'une information incomplète au moyen de la notion de *vue-monde* dans des situations où la notion précédente d'ensemble réponse n'est pas assez précise pour traiter le raisonnement de sens commun et où il y a une multitude d'ensembles réponses. Nous ajoutons ici des opérateurs épistémiques au langage original de la logique **HT** et nous définissons une version épistémique de la logique de l'équilibre. Cette version épistémique constitue une nouvelle sémantique non seulement pour

les spécifications épistémiques de Gelfond, mais aussi plus généralement pour les programmes logiques épistémiques étendus. Enfin, nous comparons notre approche avec les sémantiques existantes et nous proposons une équivalence forte pour les théories de l'**E-HT**. Ceci nous conduit naturellement des **E-S** aux **ASP** épistémiques et peut être considéré comme point de départ pour les nouvelles extensions du cadre **ASP**.

Contents

Chapter 1

Introduction

This chapter is composed of two main parts: (1) the first part lays out the philosophy of the main problem of this dissertation, explains the significance of the problem and briefly introduces the approaches towards the solution. Having the aim of the dissertation given, (2) the second part, step by step, establishes the preliminary aspects: it contains a brief overview of **ASP**, giving specific classes of logic programs in their historical progress, and defining important concepts such as answer set and strong equivalence. Then comes the heart of this work—equilibrium logic.

Computation is made up of largely disjoint areas: programming, databases (**DB**) and artificial intelligence (**AI**). *Traditional* (*imperative*) *programming* deals with expressing in code the algorithm you plan to use, i.e., posing the problem as a program pattern and then executing it to create some output, if possible. In other words, it is basically focused on describing how a program operates. *Procedural programming* is kind of imperative programming in which the program is built from one or more procedures. *Database* refers to an organised collection of data which is created to operate large quantities of information by inputting, storing, retrieving and managing it. *Artificial intelligence*, that was coined in 1955 by John McCarthy, is the study and design of intelligent agents which are systems that perceive their environment and take actions in order to maximise their chances of success. **AI** was founded on the claim that a central property of humans, i.e., intelligence can be precisely described so that a machine can be made to simulate it. The driving force behind logic programming (**LP**) is the idea that a single formalism suffices for both logic and computation. So, **LP** unifies different areas of computation by exploiting the greater generality of logic.

Declarative programming, in contrast to imperative and procedural programming, specifies the problem having the program figure out itself a way to produce a solution. Hence, programs themselves describe their desired results without explicitly listing commands or steps that must be performed. For example, logi-

cal programming languages are characterized by a declarative programming style. However, some logical programming languages, such as Prolog and database query languages, like SQL, while declarative in principle, also support a procedural style of programming. In logical programming languages, programs consist of logical statements, and the program executes by searching for proofs of the statements. To sum up, *declarative problem solving* thus focuses on what the program should accomplish without prescribing how to do it in terms of sequences of actions to be taken.

In a general aspect, *answer set programming* (or **ASP**, for short) as a recent problem solving approach relates logic programming (**LP**) to declarative problem solving through *stable models*, so **ASP** originates from the relation between two questions: "what is the problem?" versus "how to solve a given problem?".

A problem solving procedure in **ASP** operates as follows: a problem instance I is first encoded as a (non-monotonic) logic program Π such that a solution of I is represented by a distinguished model of Π, so the solving procedure is reflected to a relation from a logic program to selecting these specified models which are computed according to a minimisation principle. To express it more clearly, in **ASP** paradigm solving a problem instance I is reduced to computing the stable models of the corresponding logic program Π_I. As a result, the set of stable models of a program Π_I is the set of all solutions to a problem instance I, and hence stable model semantics provides a multiple model semantics (Eiter [2008]). There are two programs called *grounders* and *answer set solvers* responsible for generating stable models. The use of answer set solvers in search of stable models was identified in 1999 as a new programming paradigm by Marek and Truszczyński (Marek and Truszczyński [1999]; Niemelä [1999]). The computational process employed in the design of many answer set solvers is an enhancement of the DPLL algorithm.

ASP uses Prolog-style query evaluation for solving problems. However, in **ASP** this evaluation always terminates, and never leads to an infinite loop. **ASP** is oriented towards difficult combinatorial search problems in the realm of P, NP, NP^{NP} and Σ_p^2. In particular, it allows for solving all search problems of NP (and, NP^{NP}) complexity in a uniform way. To sum up, **ASP** is an approach to declarative problem solving that combines a rich, yet simple modeling language with high performance solving capacities.

ASP is closely aligned with Knowledge Representation and Reasoning (**KR**&**R**), and in particular, gets it roots from (Gebser et al. [2012]):

- (deductive) databases (**DB**)
- logic programming (with negation) (**LP**)
- (logic-based) knowledge representation (**KR**) and nonmonotonic reasonig

- constraint solving (in particular and at an abstract level, it relates to satisfiability (SAT) solving and CSP)

As a result of this, **ASP** benefits from the integration of **DB**, **KR** and **SAT** techniques. It offers a concise problem representation, so provides rapid application development tools. Moreover, it handles of knowledge intensive applications including data, frame axioms, exceptions, defaults, closures, etc, and in dynamic domains like Automated Planning. So, we could briefly display it as follows:

$$\textbf{ASP} = \textbf{DB} + \textbf{LP} + \textbf{KR} + \textbf{SAT}.$$

ASP has a two-sided language: viewed as a high-level language, it expresses problem instances as sets of facts, encodes classes of problems as sets of rules, and proposes the solutions through their stable models of facts and rules. On the other hand, viewed a low-level language, it compiles a problem into a logic program, and solves the original problem by solving its compilation.

ASP is a relatively new paradigm of declarative programming which has a history of at most 40 years. Although the term was first coined by Lifschitz in 1999 (Lifschitz [1999a, 2002]), the same approach was simultaneously proposed by some other researchers (Marek and Truszczyński [1999]; Niemelä [1999]), and explained in full detail by Baral in Baral [2003]. The impact of the theory has been immense. One reason is that it is hugely versatile since it embraces many emerging application areas and theoretical aspects. Especially in nonmonotonic reasoning, it has proved to be a successful approach. As an outgrowth of research on the use of nonmonotonic reasoning in knowledge representation, it has been regarded as an appropriate tool, particularly, in knowledge intensive applications. The efficient implementations of **ASP** has become a key technology for declarative problem solving in the AI community (Gebser et al. [2007, 2009]). In recent decades many important results have been obtained from a theoretical point of view, such as the definitions of new comprehensive semantics like equilibrium semantics (Pearce [1996, 2006]) or the proof of important theorems like strong equivalence theorems Lifschitz et al. [2001]. These theoretical and practical results show that **ASP** is central to various approaches in non-monotonic reasoning.

Equilibrium logic was first introduced by Pearce (Pearce [1996]) in the Spring of 1995, mainly as a new logical foundation for **ASP**, and since then it has been widely investigated and further developed by several researchers. Equilibrium logic gets this popularity, in particular, since it captures the answer set semantics (Gelfond and Lifschitz [1988b, 1990]) of **ASP** and proposes an easier approach of minimal model reasoning rather than fixed point approach followed by answer set semantics. Moreover, equilibrium logic extends the restricted syntax character of ASP programs that is based on rules. In general terms, a rule is a structure of the form '$head \leftarrow body$' in which head and body are supposed to be a list of

syntactically simple expressions, such as propositional variables or literals. The backward arrow '$\leftarrow$' is read as "if". However, a rule is able to express many kinds of information. Initially, the language of **ASP** was that of general logic programs whose rules have the form '$p_1 \leftarrow (p_2\,,\,\ldots\,,\,p_n)\,,\,(\mathit{not}\,p_{n+1}\,,\,\ldots\,,\,\mathit{not}\,p_k)$' where the p_i's are propositional variables. Successively, the language was extended to handle integrity constraints, epistemic disjunction, and a second negation operator called strong negation. Equilibrium logic further enriches even these syntax extensions to a general propositional language and provides a simple minimal model characterisation of answer sets on the basis of a well-known nonclassical logic called here-and-there (**HT**) logic. It therefore provides a natural generalisation of logical consequence under answer set semantics as well as a mathematical foundation for **ASP**.

To have extended the language of **ASP** was however not the only reason to be interested in equilibrium logic. Beyond this feature, since equilibrium logic has generalised reasoning with answer sets, *a fortiori* it is closely associated with more general knowledge representation formalisms such as default logic, autoepistemic logic or modal nonmonotonic systems. Equilibrium logic provides:

- a general methodology for building nonmonotonic logics (Pearce and Uridia [2011]);
- a logical and mathematical foundation for **ASP**-type systems, enabling one to prove useful metatheoretic properties such as strong equivalence;
- further means of comparing **ASP** with other approaches in nonmonotonic reasoning.

So, it indeed deserves to be further developed to provide a general framework for many forms of nonmonotonic reasoning.

In the last decades of the twentieth century, the language of **ASP** ($\mathcal{L}_{\mathbf{ASP}}$, for short) has also been extended to even a much richer syntax by the representation of epistemic modalities to the newest version of logic programs, resulting in Gelfond's epistemic extension of **ASP** called *epistemic specifications* (Gelfond [1991, 1994, 2011]). Afterwards, the language $\mathcal{L}_{\mathbf{ASP}}$ has been further enriched by some new concepts such as nested expressions (Lifschitz et al. [1999]), ordered disjunction (Brewka et al. [2004b]), etc. Moreover, some additional features have been included into $\mathcal{L}_{\mathbf{ASP}}$, which are mainly non-propositional constructions, such as choice rules (Brewka et al. [2004a]), weight constraints and aggregates (Pearce [2006]). The outgrowth in this field of research has forced equilibrium logic to get oriented in different directions, mainly to problems arising in the foundations of **LP** under answer set semantics. This driving force has recently caused a move in this research area, for instance:

- Weight constraints can be represented as nested expressions (Ferraris and Lifschitz [2005]).
- The semantics of aggregates represented by rules with embedded implications in **ASP** has been captured (Ferraris [2005]).
- Logic programs with ordered disjunction (Brewka et al. [2004b]) have been captured (Cabalar [2010]).
- Epistemic specifications (Gelfond [1994]) have been captured (Chen [1997]; Wang and Zhang [2005]).

However, there is still not an ultimate success that has catched up all current progress in **ASP**. On a broad scale, our work is motivated out of such a reason.

As a consequence of what is mentioned above, many new applications in **AI** drive us to extend the original language of equilibrium logic by some new concepts such as the representations of modalities, actions, ontologies or updates so as to obtain a more comprehensive framework. Based on a tradition that was started by Alchourrón, Gärdenfors and Makinson (Alchourrón et al. [1985]) and also by Katsuno and Mendelzon (Katsuno and Mendelzon [1992]), several researchers have proposed to enrich **ASP** (and hence, equilibrium logic) by operations allowing to update or revise a given ASP program through a new piece of information (Eiter et al. [2002]; Slota and Leite [2012a,b]; Zhang and Foo [2005]). However, the resulting formalisms have been quite complex so far, and we think that it is fair to say that it is difficult to grasp what the intuitions should be like under these approaches. Therefore, this work aimes at giving more modest and neat extensions of **ASP**. Besides, concerning the extensions of equilibrium logic, only a few approaches exist up to now. Among them there are essentially the investigation of monotonic formalisms underlying equilibrium logic by means of the concepts of contingency (Fariñas del Cerro and Herzig [2011b]), of modal operators quantifying over here and there worlds in the definition of an equilibrium model (Fariñas del Cerro and Herzig [2011a]), and of temporal extensions of equilibrium logic (Aguado et al. [2008]; Cabalar and Demri [2011]). Hence, this work also tries to cover these already existing formalisms.

This dissertation basically addresses the problem of logical foundations of **ASP**. More specifically, we search for the monotonic formalisms underlying equilibrium logic, and propose extensions of equilibrium logic by some modal concepts. Throughout this line of work, we first approach the subject from a perspective of capturing equilibrium logic in a modal logic framework. We have proposed a monotonic modal logic called **MEM** that is able to capture equilibrium logic. Then, we continue our work by extending the original language of equilibrium logic by dynamic operators which provide updates of HT models because we believe that

the investigation of modal extensions of **HT** logic and equilibrium logic is a necessary and natural starting point for characterising extensions of ASP with modal operators. In that perspective, we also tackle the same problem with epistemic operators and propose an an epistemic extension of equilibrium logic which is so able to suggest a new logical semantics not only for Gelfond's epistemic specifications, but also for more general nested epistemic logic programs Wang and Zhang [2005]. However, we have no ultimate success with the latter work and it needs to be further progressed. A parallel thorough work has been lately carried out by Cabalar *et al.* (Aguado et al. [2008]; Cabalar and Demri [2011]; Diéguez Lodeiro [2015]), so as future work we plan to compare the resulting formalisms with the already existing temporal extensions of equilibrium logic. Although our work seems to be specific and technical for the non-logician relying solely on the aim of extending equilibrium logic, in future we will try to make our work exciting for researchers who are less familiar with equilibrium logic by revealing the connections of our work to **ASP** and some nonmonotonic formalisms such as autoepistemic logic, default logic, **S4F**, etc.

In the following section we give a brief overview of **ASP**: we identify specific classes of logic programs and some new constructs in **ASP** by recalling a historical perspective of the stable model semantics as well as introducing the important concept of strong equivalence.

1.1 What is Answer Set Programming (ASP) ?

The language of **ASP** is the same as the language of **LP** in essence, except that the latter is a *programming* language, while the former is a *logical* (or, *purely declarative*) language. The semantics of **ASP** is based on the *stable model* (currently, *answer set*) semantics over **LP** (Gelfond and Lifschitz [1988a]). The new terminology of answer set instead of stable model was first proposed by Lifschitz (Lifschitz [1999b]) as a generalization of the latter for logic programs with strong negation. The concept of stable model applies ideas of autoepistemic logic (Moore [1985a]) and default logic (Reiter [1980]), which are commonly accepted formalisms for knowledge representation, to the analysis of negation as failure (NAF, for short). As a result, **ASP** includes all applications of answer sets to knowledge representation (Baral [2003]; Gelfond [2008]).

In **LP** history, the 80s and the 90s have happened to be the scene of a war on a suitable semantics allowing for an understanding of programs with NAF. Stable model semantics appeared on this scene in the late 80s. It was first proposed by Gelfond and Lifschitz (Gelfond and Lifschitz [1988b]), and since then during a ten year period, the **LP** community approached the concept hesitantly mainly because it seemed that the stable model semantics is peculiarly oriented towards

NAF, that is to say, it is particularly useful for just interpreting this operator. Moreover, some researchers supported this intuition with some formal evidence. At the same time, they thought that the stable model semantics did not fit into a standard paradigm of **LP** languages. There was a great schism: single model versus multiple model semantics. While standard approaches assign to a logic program a single *intended* model, stable model semantics lacks of a single intended model, assigning to a program a family (possibly empty) of *intended* models. Moreover, the abstract properties satisfied by the consequence relation associated with stable model semantics could not compete with its rivals since while it fails some desirable properties like cumulativity and rationality, they hold for its rivals such as well founded semantics. For a while, this was regarded by some critics as a negative feature of the semantics. Later, with the rise of efficient answer set solvers and the practical viability of **ASP** as a **KR** and programming paradigm, such criticisms were no longer effective. Last but not least, they were recognised as inadequate in resolution-based proof search. As a consequence of these difficulties in reconciling the stable model semantics with a traditional paradigm of **LP**, in the 90s, the stable model semantics received relatively less attention from the **LP** community than other semantics proposed for programs with NAF such as perfect model semantics for stratified programs (Bachmair and Ganzinger [1991]; Maher [1993]; Przymusinska and Przymusinski [1988]; Przymusinski [1988a]) and well founded semantics for general logic programs (Van Gelder et al. [1991]).

The negative fame of stable models has turned into a reasonable, but very belated fortune just in the beginning of 21th century. In time it has been understood that instead of proofs, models in the form of stable models (more generally, answers sets) provide informative interesting solutions, so it has been widely searched for new techniques computing models from then on.

1.1.1 Logic programs and answer sets: general definition

Formulas in **LP** have been initially built from a set of *propositional variables* (or *atoms*) $\mathbb{P} = \{p, q, \ldots\}$ and the 0-place connectives $\top$ and $\bot$ using *negation as failure* (*not*) and conjunction (,). As mentioned in the following sections, later on this language has been further extended by *epistemic disjunction* (*or*) and a second kind of negation called *strong negation* ($\sim$) which expresses the direct or explicit falsity of a propositional variable. A detailed discussion about strong negation can be found in Appendix B.

In general terms, a *literal* is referred to as a propositional variable or its negation, and a *clause* is a finite disjunction of literals. Particularly, in our context, we mainly call a literal, a propositional variable or strongly negated propositional variable, and a clause an (finite) epistemic disjunction of such literals. However, as long as it is clear from the context, we continue using the same terms when

different kinds of negations and disjunctions are used, so the reader should not get confused when we sometimes use the term 'clause' in the sense of 'rule' (of a logic program). Literals can be divided into two parts: if a literal is simply a propositional variable then we call it a *positive* literal, otherwise a *negative* literal. We will usually denote a literal by l and its complementary, i.e., the literal opposite in sign to l by $\bar{l}$. To express it more clearly, if $l = p$ then $\bar{l} = \sim p$ and if $l = \sim p$ then $\bar{l} = p$. Moreover, we denote by $\mathbb{P}^+$ the set of all positive literals which equals at the same time the set of propositional variables $\mathbb{P}$ and by $\mathbb{P}^-$ the set of all their strong negations. Moreover, we call *Lit* the set of all ground (without variables) literals. In other words, *Lit* is the union of $\mathbb{P}^+$ and $\mathbb{P}^-$, i.e., $Lit = \mathbb{P}^+ \cup \mathbb{P}^-$. In this dissertation, we mainly restrict ourselves to the propositional case, so we mostly deal with ground literals, and omit quantifiers in the language of **LP** ($\mathcal{L}_{\mathbf{LP}}$) except some well-known examples by Gelfond, directly quoted in Section 1.4.

Throughout this work, we denote the set of propositional variables and the set of literals occurring in a formula φ respectively by $\mathbb{P}_\varphi$ and Lit_φ This notation is then generalised to theories. We will follow the same notation also for rules and logic programs.

The negation as failure (alias, *default negation*) *not* has the following intuitive meaning: *not* φ stands for "φ is false by default". In a more precise explanation, it means that when there is no evidence for adopting φ, or when we have no acceptable support that provides a justification for φ, we accept φ to be false. Negation as failure (NAF) has been an important feature of **LP** since the earliest days of both Planner and Prolog. NAF is a non-monotonic inference rule in **LP**, used to derive *not* φ (i.e., that φ is assumed not to hold) from failure to derive φ. Ray Reiter investigated NAF in the context of a first order database D, interpreting it as the *closed world assumption* (CWA) that the negation *not* p of a ground predicate p holds in D if there is no proof of p from D (Reiter [1978]). However, the semantics of NAF remained an open issue until Keith Clark (Clark [1978]) who was the first to investigate this operator in the context of logic programs. Loosely speaking, his solution was to interpret rules of a logic program P in 'if-and-only-if' form called the *completion* (sometimes, the *predicate completion* or the *Clark completion*) of P. More recently, Michael Gelfond showed that it is also possible to interpret *not* p literally as "p can not be shown", "p is not believed" or "p is not known to be true" as in autoepistemic logic (Gelfond [1987]). The autoepistemic interpretation was developed further by Gelfond and Lifschitz (Gelfond and Lifschitz [1988b]) and has become a starting point for **ASP**.

A rule r is an ordered pair $(body(r), head(r))$ which has an explicit representation of the form

$$r = head(r) \leftarrow body(r) \tag{1.1}$$

where $head(r)$ and $body(r)$ are respectively a (epistemic) disjunction and a conjunc-

tion of arbitrary formulas. We call $head(r)$ and $body(r)$ a set of possible conclusions and a set of conditions separately in the given order. In particular, $body(r)^+$, which is the *positive part* of $body(r)$, refers to conjuncts of $body(r)$ not containing NAF. We alternatively call $body(r)^+$ the *premises* of r. Similarly, $body(r)^-$, which is the *negative part* of $body(r)$, refers to conjuncts of $body(r)$ preceded by NAF. We call this part *constraints* of r. Particularly, when $body(r)$ is free of conditions, i.e., contains no conjuncts at all then we consider $body(r) = \top$ since $\bigwedge \emptyset = \top$. In this case, we identify rule (see Definition 1.1) with $head(r) \leftarrow$ (or simply with the formula $head(r)$), and it refers to a *true statement* (regardless of a condition) or a *fact*. On the other hand, if $head(r)$ contains no disjuncts then we take $head(r) = \bot$ since $\bigvee \emptyset = \bot$, and identify the resulting structure with $\leftarrow body(r)$. This structure is usually called a *constraint* (or sometimes a *goal statement*). Finally, we call a trivial rule, which is in the form '$\top \leftarrow \top$' or '$\bot \leftarrow \bot$', an *empty rule* and denote it by $\top$. It is useful to name it just because we will implicitly come across it while producing reducts (see, for example, definitions R2.1 and R3.1).

A rule r has the following explicit form:

$$head(r) \leftarrow body(r)^+ \ , \ body(r)^- \tag{1.2}$$

in which: for an arbitrary formula φ_i $(1 \leq i \leq k)$ and for $0 \leq m \leq n \leq k$, we have

$$\begin{array}{lcl} head(r) & = & \varphi_1 \, or \, \ldots \, or \, \varphi_m \\ body(r) & = & (\varphi_{m+1} \, , \, \ldots \, , \, \varphi_n) \, , \, (not \, \varphi_{n+1} \, , \, \ldots \, , \, not \, \varphi_k) \\ body(r)^+ & = & \varphi_{m+1} \, , \, \ldots \, , \, \varphi_n \\ body(r)^- & = & not \, \varphi_{n+1} \, , \, \ldots \, , \, not \, \varphi_k. \end{array}$$

Some researchers in the field also use a set notation and represent $head(r)$ and $body(r)$ simply as sets:

$$\begin{array}{lcl} head(r) & = & \{\varphi_1, \ldots, \varphi_m\} \\ body(r) & = & \{\varphi_{m+1}, \ldots, \varphi_n, \, not \, \varphi_{n+1}, \ldots, \, not \, \varphi_k\} \\ body(r)^+ & = & \{\varphi_{m+1}, \ldots, \varphi_n\} \\ body(r)^- & = & \{not \, \varphi_{n+1}, \ldots, \, not \, \varphi_k\} \end{array}$$

but since it is not very user-friendly, we prefer the former notation. However, the reader should not get confused when we say, for example, "$body(r) = \emptyset$" or "$headr(r) = \emptyset$" to refer to a fact (disjunctions of formulas) or a constraint respectively. Similarly, we sometimes write "$body(r) = head(r) = \emptyset$" to refer to an empty rule, and even "$body(r)^- = \emptyset$" to refer to a positive program rule (see the next paragraph).

A *logic program* is a finite set of rules. In particular, a program Π of rules r is called *positive* if $body(r)^- = \top$ (in other words, if $body(r)^-$ contains no conjuncts)

for all its rules r. For instance, every Horn program (see Subsection 1.1.2.1) is a positive logic program.

The general representation (Definition 1.2) of rules comprises all forms of logic program rules that will be discussed in the following subsection. For simplicity, we sometimes prefer to indicate this general form of rules in compact terms by

$$\bigvee_{1}^{m} \varphi_i \leftarrow \Big(\bigwedge_{m+1}^{n} \varphi_i \Big) \wedge \Big(\bigwedge_{n+1}^{k} \mathit{not}\, \varphi_i \Big), \quad \text{for } 0 \leq m \leq n \leq k \tag{1.3}$$

where '$\wedge$' and '$\vee$' respectively refer to our formal (official) symbols ',' and '*or*' of $\mathcal{L}_{\mathbf{ASP}}$. Except this compact representation where their differences from classical '$\wedge$' and '$\vee$' are clear from the context, we will stay loyal to our notational conventions. However, in the literature, such notations are used interchangeably in order to stress the translation between program rules and corresponding propositional formulas. In a broader sense, one can come across '$\leftarrow$', '*or*' (;), '*not*' and ',' viewed respectively as '$\rightarrow$', '$\vee$', '$\neg$' and '$\wedge$'. Moreover, some programs are defined in first-order forms, including free variables, and even quantifiers. Programs including rules seemingly with free variables are usually treated as shorthands for the set of their ground instances. So, in one sense free variables in a logic program are (usually universally) quantified bound variables given succinctly.

Given $X \subseteq \mathit{Lit}$, we write $X \models \varphi$ if X satisfies φ and we write $X =\!| \varphi$ if X falsifies φ. The truth and the falsity conditions of the language $\mathcal{L}_{\mathbf{ASP}}$ are defined inductively by:

$$\begin{array}{lll}
X \models l & \text{if} & l \in X, \text{ for } l \in \mathit{Lit} \\
X \models \top & & \\
X \models (\varphi\,,\, \psi) & \text{if} & X \models \varphi \text{ and } X \models \psi \\
X \models (\varphi\, \mathit{or}\, \psi) & \text{if} & X \models \varphi \text{ or } X \models \psi \\
X \models \mathit{not}\, \varphi & \text{if} & X \not\models \varphi \\
\\
X =\!| p & \text{if} & {\sim}p \in X, \text{ for } p \in \mathbb{P} \\
X =\!| {\sim}p & \text{if} & p \in X, \text{ for } p \in \mathbb{P} \\
X =\!| \bot & & \\
X =\!| (\varphi\,,\, \psi) & \text{if} & X =\!| \varphi \text{ or } X =\!| \psi \\
X =\!| (\varphi\, \mathit{or}\, \psi) & \text{if} & X =\!| \varphi \text{ and } X =\!| \psi \\
X =\!| \mathit{not}\, \varphi & \text{if} & X \models \varphi.
\end{array}$$

In fact, we could give the falsity definition concisely in terms of a truth condition: '$X =\!| \varphi$ if $X \models {\sim}\varphi$', but in $\mathcal{L}_{\mathbf{ASP}}$ strong negation just precedes propositional variables, resulting in a literal. So, we prefer staying loyal to the original language. Satisfaction and falsification definitions are generalised to rules and programs as follows: X satisfies (falsifies) a program rule r, written $X \models r$ ($X =\!| r$), if $X \models \mathit{head}(r)$ ($X =\!| \mathit{head}(r)$) whenever $X \models \mathit{body}(r)$ ($X =\!| \mathit{body}(r)$). Moreover, we

write $X \models \Pi$ when X satisfies a program Π, i.e., $X \models r$ for every $r \in \Pi$. Similarly, we write $X =| \Pi$ when X falsifies Π, i.e., $X =| r$ for some $r \in \Pi$. One should note that a set $X \subseteq Lit$ does not necessarily falsify a rule (or a program) if it does not satisfy it. An easy counterexample is given by $X = \emptyset$ and p: $\emptyset \not\models p$ since $p \notin X$, but $\emptyset =| p$ does not hold either since $\sim p \notin X$. A *theory* Γ is a finite set of propositional formulas. The definitions given above can be adapted to propositional formulas and theories straightforwardly.

After having received a wide acceptance by the **LP** community, the stable model concept (Gelfond and Lifschitz [1988b]) has been followed as the semantics of logic programs. However, shortly after the strong negation is added into the language of logic programs, the semantics of logic programs is turned into a more general concept called answer sets (Gelfond and Lifschitz [1990]). The main difference between these two concepts are as follows: an answer set is basically a set of ground literals, and it has been first defined for extended logic programs (see Subsection 1.1.2.2), i.e., programs containing a second negation called strong negation. However, a stable model has been defined for general logic programs (see Subsection 1.1.2.2), i.e., the simplest logic programs containing just one negation (NAF), and is simply a set of propositional variables. In other words, a stable model is a valuation, that is, a classical model of a program viewed as a set of propositional formulas with incomplete information of the world. In that respect, a stable model is simply a minimal (in the sense of the smallest proper subset) incomplete (3-valued) Herbrand model. However, the converse does not hold in general: not every minimal Herbrand model is stable. This means that stable model concept is defined according to a fixed point property that involves minimality conditions, but is not a classical form of minimal model reasoning. On the other hand, it does correspond, via a suitable translation, to default and autoepistemic reasoning. For simplicity we first limit our attention to programs without strong negation, and introduce stable model concept . However, answer set concept is defined in analogy to stable models, so it is easy to extend the definitions given below to programs containing strong negation.

The stable model semantics below defines when a classical model of a propositional formula is considered 'stable'. Two different definitions of stable models are given and both definitions are equivalent, so they lead to exactly the same stable models.

According to the first definition (Ferraris [2005]), we treat a ground rule as a propositional formula and a ground program as a theory. The reduct φ^X of a propositional formula φ relative to a set $X \subseteq \mathbb{P}$ is the formula obtained from φ replacing each maximal subformula that is not satisfied by X by $\bot$. In a formal way, the reduct φ^X is defined recursively as follows:

R1.1 if $X \not\models \varphi$, then $\varphi^X = \bot$,

R1.2 if $X \models p$ (i.e., $p \in X$), for $p \in \mathbb{P}$, then $p^X = p$, and

R1.3 if $X \models \varphi \otimes \psi$, then $(\varphi \otimes \psi)^X = \varphi^X \otimes \psi^X$,

where $\otimes$ refers to a binary connective. Then, we set $\Gamma^X = \{\varphi^X \ : \ \varphi \in \Gamma\}$ for a theory Γ. Having the reduct definition given, we say that X is a stable model of Γ if X is minimal among the sets satisfying Γ^X. In this context, the minimality of X is understood in the sense of set inclusion: no proper subset of X satisfies Γ^X. Clearly, every stable model of a theory Γ (in particular, of a formula φ) according to this definition is a model of Γ: indeed if X does not satisfy Γ then $\bot$ belongs to Γ^X. For instance, we can identify a program Π:

$$\begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & not\, r \\ s\, or\, r & \leftarrow & p \end{array}$$

with the theory $\{q \rightarrow p, \neg r \rightarrow q, p \rightarrow s \vee r\}$ or even with the formula

$$\varphi = (q \rightarrow p) \wedge (\neg r \rightarrow q) \wedge (p \rightarrow s \vee r).$$

Then, to check that $\{p, q, s\}$ is a stable model of φ, we take the reduct

$$\varphi^{\{p,q,s\}} = (q \rightarrow p) \wedge (\neg \bot \rightarrow q) \wedge (p \rightarrow s \vee \bot),$$

or equivalently $(q \rightarrow p) \wedge q \wedge (p \rightarrow s)$, and show that $\{p, q, s\}$ is minimal among its models.

While Ferraris' definition aims at eliminating unsatisfied subformulas, in contrast, according to the second definition (Lifschitz et al. [2001]), the key point is to eliminate the NAF operator occurring in a program. To this end, the *reduct* Π^X of a program Π with respect to a set $X \subseteq \mathbb{P}$ is given through replacing every maximal occurrence of a formula of the form $not\, \varphi$ in Π (that is, every occurrence of $not\, \varphi$ that is not in the range of another not) with

R2.1 $\bot$ if $X \models \varphi$,

R2.2 $\top$ if $X \not\models \varphi$.

The stable model definition is first given for positive programs, i.e., programs containing no negation operator (neither NAF nor strong negation). A set $X \subseteq \mathbb{P}$ is *closed* under a positive program Π if $X \models head(r)$ whenever $X \models body(r)$ for every rule r (see Definition 1.1) in Π. Therefore, the closure concept refers to the satisfaction concept. More generally, X being closed under Π amounts to X being a classical model of Γ_Π where Γ_Π is the theory that corresponds to the program Π. Then, we denote by $Cn(\Pi)$ the $\subseteq$-smallest (minimal) sets of propositional variables

that is closed under a positive program Π. Likewise, $Cn(\Gamma_\Pi)$ corresponds to the $\subseteq$-smallest classical models of Γ_Π when Π is transformed into a theory. However, one should note that in principle there could be several such sets. For example, for the positive program $\Pi = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & \\ s\,or\,r & \leftarrow & p \end{array} \right.$, $Cn(\Pi)$ corresponds to two sets: $S_1 = \{p, q, s\}$ and $S_2 = \{p, q, r\}$. We define a *stable model* of a positive program Π as a minimal set closed under Π, and this is nothing but $Cn(\Pi)$ itself. Hence, both S_1 and S_2 are the stable models of the program Π. At this point, it is useful to notice that $\Pi^{S_1} = \Pi^{S_2} = \Pi$.

We call two positive programs Π and Π' *equivalent* in the sense of their models when Γ_Π and Γ'_Π have exactly the same classical models and we call them *equivalent* under the semantics of positive programs (i.e., in the sense of their stable models) when $Cn(\Pi)$ and $Cn(\Pi')$ both correspond to the same sets.

The restricted definition of a stable model for a positive program is then generalised to a program containing only NAF (but not strong negation) as follows: given a program Π, we first eliminate all occurrences of *not* from Π relative to $X \subseteq \mathbb{P}$ and form the reduct Π^X, then we say that X is a *stable model* for Π if X is a stable model for the reduct Π^X, in other words, if $Cn(\Pi^X)$ corresponds to X. The latter is known as the *fixed point property*. Slightly changing the example above and reexamining it as $\Pi = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & not\,r \\ s\,or\,r & \leftarrow & p \end{array} \right.$, we get $\Pi^{\{p,q,s\}} = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & \top \\ s\,or\,r & \leftarrow & p \end{array} \right.$. Clearly, $\Pi^{\{p,q,s\}}$ has two minimal models: $\{p, q, s\}$ and $\{p, q, r\}$. However, just the former is stable since it satisfies the fixed point property, but not the latter. Indeed, $\{p, q, r\}$ is not a minimal model of $\Pi^{\{p,q,r\}} = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & \bot \\ s\,or\,r & \leftarrow & p \end{array} \right.$ which is equivalent to $(\Pi^{\{p,q,r\}})' = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ s\,or\,r & \leftarrow & p \end{array} \right.$ in the sense of its models. This is clearly because some strict subsets of $\{p, q, r\}$ such as $\{p, r\}$, $\{r\}$ and $\emptyset$ are also closed under this reduct.

As a next example, we claim that $\{p, q\}$ is a stable model for the program Π_1:

$$\begin{array}{l} q \leftarrow \\ p \leftarrow s\,,\, not\,q \\ p \leftarrow q\,,\, not\,r. \end{array}$$

Indeed, $\Pi_1^{\{p,q\}} = \left\{ \begin{array}{l} q \leftarrow \\ p \leftarrow s\,,\, \bot \\ p \leftarrow q\,,\, \top \end{array} \right.$ which is equivalent to $(\Pi_1^{\{p,q\}})' = \left\{ \begin{array}{l} q \leftarrow \\ \top \\ p \leftarrow q \end{array} \right.$ and

even, to $(\Pi_1^{\{p,q\}})'' = \left\{ \begin{array}{l} q \leftarrow \\ p \leftarrow \end{array} \right.$ in the sense of its models. Thus, $\{p, q\}$ is clearly the minimal set closed under this reduct. As this example justifies, it is easy to see that the reduct definition given above can be equivalently interpreted as follows: given a logic program Π, the *reduct* Π^X relative to $X \subseteq \mathbb{P}$ is obtained from Π by deleting

R2′.1 each rule having $not\, p$ (for $p \in \mathbb{P}$) in its body with $p \in X$, and

R2′.2 all occurrences of $not\, p$ (for $p \in \mathbb{P}$) in the bodies of the remaining rules.

Hence, the alternative definition of Π^X can be formulated as:

$$\Pi^X = \{head(r) \leftarrow body(r)^+ \ : \ r \in \Pi \text{ and } \mathbb{P}_{body(r)^-} \cap X = \emptyset\}.$$

If we, once again, work over the same example where $\Pi = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & not\, r \\ s\ or\ r & \leftarrow & p \end{array} \right.$, we get $\Pi^{\{p,q,s\}} = \left\{ \begin{array}{rcl} p & \leftarrow & q \\ q & \leftarrow & \\ s\ or\ r & \leftarrow & p \end{array} \right.$. Hence, the same result immediately follows this time. We will follow mainly this alternative reduct definition, in which the results are more explicitly given, for further examples on stable models. One should note that the reduct definition only aims at eliminating the *not* operator, so it is directly oriented to the propositional variables preceded by *not* in the body of rules.

Stable models of a traditional logic program (i.e., programs without NAF and nested expressions in the head) have the following properties: a stable model of a program Π should always be a subset of $\mathbb{P}_\Pi$ which means that it is finite. Moreover, only the propositional variables occurring in the head of rules of Π can appear in a stable model. Therefore, if a program consists of constraints only, then the unique stable model of this program is the empty set in case there is one. For instance, the empty set is the unique stable model of the program $\Pi = \left\{ \begin{array}{l} \leftarrow p\,,\ not\, q \\ \leftarrow q\,,\ not\, p \end{array} \right.$. However, another program $\Pi' = \left\{ \begin{array}{l} \leftarrow not\, p \\ \leftarrow not\, q \end{array} \right.$ of this kind has no stable models. If S_1 and S_2 are two stable models of the same logic program Π then S_1 is not a proper subset of S_2 (i.e., neither $S_1 \subset S_2$ nor $S_2 \subset S_1$ holds) due to the minimality condition in the definition of stable models: every stable model of a logic program Π is minimal among the models of Π relative to set inclusion. This property is known as the *antichain property*. So, the set of stable models of a program is an antichain. Thus, if '$\emptyset$' is a stable model of a program then it must be unique.

Both of the above-mentioned definitions of a stable model can then be generalised to the definition of an answer set in a natural way, but this time we consider

a set S' of literals (i.e., a subset of *Lit*) rather than a set S of propositional variables (i.e., a subset of $\mathbb{P}$) and the rest is the same except one case where the former contains a pair of complementary literals (i.e., p and $\sim p$ together for $p \in \mathbb{P}$): if S' contains a pair of complementary literals then $S' = Lit$. This is because p and $\sim p$ together lead to a contradiction (see Appendix B.1.1). An answer set of a program meets its stable model when the program does not include strong negation: the answer set notion is a conservative extension of the stable model notion. Therefore, answer set semantics comprises the semantics of all forms of logic programs although it is mainly defined for extended logic programs. However, in fact there is an essential difference between these two concepts: the absence of $p \in \mathbb{P}$ in a stable model of a general logic program Π refers to the fact that "p is false", yet the absence of $p \in \mathbb{P}$ (and expectedly of $\sim p \in Lit$) in an answer set of the same program means that nothing is known about p. That is why we will mainly follow the latter as the formal semantics of **ASP** henceforth.

An answer set X is said to be *consistent* (or *coherent*) if it does not contain a pair of complementary literals, otherwise it is *inconsistent* and equals *Lit*. In other words, the only acceptable inconsistent answer set is *Lit* itself. At the same time, *Lit* is the only infinite answer set possible because all others are finite. Different from answer sets, a stable model is always finite, and $\mathbb{P}$ can never be a stable model of a program.

A program Π is said to be *contradictory* if it has a unique answer set and this set equals *Lit*, otherwise it is *noncontradictory*. A program Π is said to be *inconsistent* (or *incoherent*) if it is either contradictory or it does not have any answer sets at all, otherwise it is *consistent* (or *coherent*). One should note that a program is regarded as consistent when it has at least one consistent answer set. For instance, the programs $\Pi_1 = \left\{ \; p \leftarrow not\ p \right.$ and $\Pi_2 = \left\{ \begin{array}{rl} p & \leftarrow \\ \sim p & \leftarrow \end{array} \right.$ are inconsistent respectively because the former has no answer sets (see Example G.3 for the verification) and the latter is a self-evident contradictory program. However, the program $\Pi_3 = \left\{ \begin{array}{rl} \sim p & \leftarrow \\ p\ or\ q & \leftarrow \end{array} \right.$ is consistent: it has two answer sets, $\{q, \sim p\}$ and *Lit*, but the former is consistent. Today's version of the language ($\mathcal{L}_{\mathbf{ASP}}$) differs in this sense: the set of all literals *Lit* is no longer considered the answer set of a program containing contradictory rules. For instance the program Π_2 is now said to have no answer set, and Π_3 is said to have a unique answer set $\{q, \sim p\}$.

As with stable models, a program Π cannot have two answer sets such that one is (strictly) included in another because answer sets of a program are also designed according a minimisation criterion in the sense of set inclusion. To see this fact formally, we assume for a contradiction that a program Π has two different answer sets, say S and S', such that $S \subset S'$. First we remember that $S, S' \subseteq Lit_{\Pi}$, then

it is clear that $\Pi^{S'} \subseteq \Pi^{S}$: while producing the reduct, S and S' both make the same effect on Π related with *not* l appearing in Π such that $l \in S \cup (\mathbb{P} \setminus S')$, and related with the rest such that $l \in S' \setminus S$, while S deletes the formula *not* l only, S' deletes the whole rule. Since S and S' are answers set of Π, we have $Cn(\Pi^{S'}) = S'$ and $Cn(\Pi^{S}) = S$ by definition. Hence, $S' \subseteq S$ because $\Pi^{S} \setminus \Pi^{S'}$ may add new variables into S. As a result, '$\emptyset$' and '*Lit*' are unique when they are answer sets of a program Π.

1.1.2 Specific classes of logic programs

LP unifies different areas of computing by exploiting the greater generality of logic. It does so by building upon and extending one of the simplest, yet most powerful logic imaginable, namely the logic of *Horn clauses* (Kowalski [2014]). Extending Horn clause programs with the NAF operator to reason about negative conditions was recognised from the earliest days of **LP**, and in fact **ASP** was born as a result of an effort in order to find a suitable semantics for such programs containing NAF. Hence, **ASP** programs were first introduced in the form of *general logic programs* which are extensions of Horn clause programs by NAF. Then, in the beginning of 90's, two important extensions of this standard form of logic programs were proposed by Gelfond and Lifschitz (Gelfond and Lifschitz [1990]). They redesigned logic programs allowing strong negation (yet, they used the term 'classical negation', and even some different authors called it 'explicit negation') in which program rules were built of literals rather than propositional variables. Similarly, the NAF operator was applied to some of the literals in the body. They called this newly formed version of logic programs *extended logic programs*. Moreover, Gelfond and Lifschitz introduced a new notion called *answer set* semantics as a generalisation of the stable model semantics in order to interpret programs with strong negation. Then, Gelfond and Lifschitz proposed an additional extension of the language by further allowing epistemic disjunction in the heads of program rules (Gelfond and Lifschitz [1991]). They called the resulting class of programs *disjunctive logic programs*. They also extended the notion of answer set from the case of programs with strong negation to the case of disjunctive programs. Gelfond and Lifschitz also proved that answer sets coincide with stable models in the case of general logic programs, and that they are true generalisations of stable models.

The presentation below introduces the fundamental basis and the extensions of (nonmonotonic) logic programs of increasing syntactic complexity with additional connectives in a historical order.

1.1.2.1 Horn clause basis of LP

Horn rules constitute the underlying basis of **LP** and have the form $\bigvee_{1}^{m} p_i \leftarrow \bigwedge_{m+1}^{n} p_i$ where $0 \leq m \leq 1 \leq n$ and $p_i \in \mathbb{P}$ (see Definition 1.3). Therefore, they have the following explicit representation (see Definition 1.2):

$$p_1 \leftarrow p_2 \, , \, \ldots \, , \, p_n \qquad \text{where } p_i \in \mathbb{P} \;\; \text{for } 1 \leq i \leq n. \tag{1.4}$$

However, one should keep in mind that a Horn rule, as well as the following ASP program rules, can be in the shape of a fact or a constraint. To express it more clearly, the $body(r)$ or the $head(r)$ of a Horn rule may contain no propositional variables. A *Horn program* is a finite set of Horn rules.

In the literature, the well known name for Horn rules is *Horn clauses* and they are named after the logician Alfred Horn, who studied some of their mathematical properties. The name 'clause' emphasizes that it is more common to see this rule-like form as a clause-like form with at most one positive (unnegated) literal:

$$p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \qquad \text{where } p_i \in \mathbb{P} \;\; \text{for } 1 \leq i \leq n.$$

A Horn rule r with nonempty $head(r)$ is a *definite clause* which means a disjunction of literals with exactly one positive (unnegated) literal. As a result, every definite clause is a Horn clause but not vice versa (Kowalski [2014]).

A set of definite clauses has a unique smallest model which is the intended semantics for such set of clauses. However, a set of Horn clauses has either a unique smallest model or none. For instance, while $\Pi = \{q \leftarrow p\}$ has a unique smallest model $\emptyset$, $\Pi' = \left\{ \begin{array}{ll} p & \leftarrow \\ & \leftarrow \; p \end{array} \right.$ has no model.

Horn clause programs were then extended to general logic programs letting the NAF operator appear only in the body of rules. In fact, Horn clauses were theoretically sufficient for all programming and database applications. However, they were not adequate for **AI**, most importantly because they failed to capture nonmonotonic reasoning due to lacking the NAF operator (Kowalski [2014]).

1.1.2.2 Logic programs with negation

The following programs constitute specifically the program classes of **ASP**, and each is a generalisation of the previous one, so all program forms below allow for the NAF operator.

General (Normal) logic programs

A program composed of a finite set of rules $\bigvee_{1}^{m} \varphi_i \leftarrow \left(\bigwedge_{m+1}^{n} \varphi_i \right) \wedge \left(\bigwedge_{n+1}^{k} not\, \varphi_i \right)$ (see

Definition 1.3) is called a *general* (or *normal*) *logic program* for the case $0 \leq m \leq 1 \leq n \leq k$, and when all φ_i's are simply propositional variables. Therefore, a general logic program (Gelfond and Lifschitz [1988a]; Lloyd [1987]) is a finite collection of rules (see Definition 1.2) of the following explicit form

$$p_1 \leftarrow (p_2 \, , \, \ldots \, , \, p_n) \, , \, (not \, p_{n+1} \, , \, \ldots \, , \, not \, p_k) \tag{1.5}$$

where $p_i \in \mathbb{P}$ for every $i = 1, \ldots, k$ such that $1 \leq n \leq k$. Such programs are the very basic form of ASP programs.

The name 'general' refers to the fact that such rules are more general than Horn clauses: indeed, a Horn clause can be seen as a positive general program rule r in which $body(r)^-$ is reduced to empty set. Therefore, the stable model of a positive general program Π is compatible with the smallest model of the set of clauses corresponding to Π.

The semantics followed in general **LP** is precisely the stable model semantics, but answer set concept also covers the semantics of this form of logic programs since it is a generalised version of stable models. Here are some examples.

G.1 Consider the general logic program Π_1:

$$\begin{array}{l} p \leftarrow p \\ q \leftarrow not \, p. \end{array}$$

In order to find its stable models, we search for all potential stable model candidates: $\emptyset$, $\{p\}$, $\{q\}$ and $\{p, q\}$. The smallest models of the following reducts $\Pi_1^{\emptyset} = \Pi_1^{\{q\}} = \{p \leftarrow p, \, q \leftarrow\}$, and $\Pi_1^{\{p\}} = \Pi_1^{\{p,q\}} = \{p \leftarrow p\}$ are respectively $Cn(\Pi_1^{\emptyset}) = Cn(\Pi_1^{\{q\}}) = \{q\}$ and $Cn(\Pi_1^{\{p\}}) = Cn(\Pi_1^{\{p,q\}}) = \emptyset$. However, just $\{q\}$ satisfies the fixed point property, so we eliminate the rest. Thus, the unique stable model for Π_1 is $\{q\}$.

G.2 The sets $\{p\}$ and $\{q\}$ are the (only) stable models of the program Π_2:

$$\begin{array}{l} p \leftarrow not \, q \\ q \leftarrow not \, p. \end{array}$$

Indeed, $\Pi_2^{\{p\}}$ equals $\{p \leftarrow\}$, and $\{p\}$ is clearly the minimal set closed under this reduct. Moreover, it satisfies the fixed point property. Similarly, one can show that $\{q\}$ is also a stable model for Π_2.

G.3 Finally, the program $\Pi_3 = \{p \leftarrow not \, p\}$ has no stable models: indeed among the possible candidates $\emptyset$ and $\{p\}$, none of them satisfies the fixed point property since $Cn(\Pi_3^{\emptyset}) = Cn(\{p \leftarrow\}) = \{p\}$ and $Cn(\Pi_3^{\{p\}}) = Cn(\emptyset) = \emptyset$. Moreover, $\Pi_4 = \left\{ \leftarrow not \, p \right.$, $\Pi_5 = \left\{ \begin{array}{ll} p & \leftarrow \\ & \leftarrow \; p \end{array} \right.$ and $\Pi_6 = \left\{ \begin{array}{ll} & \leftarrow \; p \\ & \leftarrow \; not \, p \end{array} \right.$ do not have any stable models either.

As shown above, a general logic program Π may have one or multiple stable models, including none. However, a 'well-behaved' program should have exactly one stable model. The existence of several stable models points out possible different interpretations about the world which can be built by a rational reasoner on the instructions from Π. Intuitively, this is why a stable model of Π cannot be strictly included in another.

An important limitation of general logic programs as a knowledge representation tool is that they do not allow us to directly deal with incomplete information. A consistent general logic program partitions the set of ground queries only into two parts: a query is answered either *yes* or *no*, depending on whether the query belongs to all its stable models or not. However, the semantics of general **LP** does not allow for a third possibility: the *unknown* answer, which corresponds to the inability to conclude *yes* or *no* (Gelfond [1989]; Gelfond and Lifschitz [1990]). For example, in the examples above while the program Π_1 (see Example G.1) is well-behaved and consistent, and answers *yes* to the query "q ?" and *no* to the query "p ?", the program Π_2 (see Example G.2) is not well-behaved, but still consistent and answers *no* to both questions although these atoms appear separately in different stable models of the latter program. However, Π_3 as well as Π_4, Π_5 and Π_6 (see Example G.3) are inconsistent and provide us no information related to these questions, but the situation here has nothing to do with the unknown answer. Finally, note that in general **LP** there is no inconsistent program in the form of 'contradictory' since its language lacks strong negation.

General logic programs are so unable to represent the incompleteness of information. This happens just because the query evaluation methods of general **LP** give the answer *no* to every query that does not succeed, automatically applying CWA to all variables. In other words, they generally provide negative information implicitly through closed world reasoning. This serious limitation has been overcome by adding another type of negation, so-called strong negation ($\sim$), besides the NAF operator (*not*) into the original language of general **LP** (Gelfond and Lifschitz [1991]). The resulting formalism is called *extended logic programming* and explained in detail in the following subsection.

To close with the complexity, testing whether a (ground) general logic program has a stable model is NP-complete.

Extended logic programs

A program composed of a finite set of rules $\bigvee_{1}^{m} \varphi_i \leftarrow \left(\bigwedge_{m+1}^{n} \varphi_i \right) \wedge \left(\bigwedge_{n+1}^{k} not\, \varphi_i \right)$ (see Definition 1.3) is called an *extended logic program* for the case $0 \leq m \leq 1 \leq n \leq k$, and when all φ_i's are literals. Hence, an extended logic program (Gelfond and Lifschitz [1990, 1991]) is a finite collection of rules of the following explicit form

(see Definition 1.2)

$$l_1 \leftarrow (l_2\,,\,\ldots\,,\,l_n)\,,\,(not\,l_{n+1}\,,\,\ldots\,,\,not\,l_k) \tag{1.6}$$

where $l_i \in Lit$ for every $i = 1, \ldots, k$ such that $1 \leq n \leq k$.

The name 'extended' emphasises the fact that the building blocks of the former language is upgraded from propositional variables to literals, having the former language enriched by strong negation, and hence that the semantics of the resulting program forms is transformed from 'stable models' to 'answer sets'. Syntactically general logic programs are a special case of extended logic programs in which literals are restricted to atoms.

Expectedly, the semantics of extended logic programs is no more the stable model semantics except that the program has no negative literals. It is now defined in terms of 'answer sets': sets of literals intuitively corresponding to possible sets of beliefs which can be built by a rational reasoner on the basis of a program Π. Hence, since an answer set S is the set of literals the agent believes to be true, any rule r with the subgoal (subformula of $body(r)$) $not\ l$ where $l \in S$ is of no use to the agent, so we produce the reduct Π^S, replacing all the rules containing such subgoals. When an answer set of Π^S coincides with S the choice of S is supposed to be 'rational'.

An extended logic program Π can be reduced to a general logic program $\widetilde{\Pi}$ by eliminating strong negation (Gelfond and Lifschitz [1991]): for every $p \in \mathbb{P}$,

i. replace each occurrence of a negative literal $\sim p$ by a fresh variable $\widetilde{p}$, which is the *positive form* in shape of the negative literal $\sim p$,

ii. keep remaining positive literals, which are already (syntactically) in positive forms.

Then, we append to $\widetilde{\Pi}$ a set of constraints called $Cons$ to guarantee the coherence: for every $p \in \mathbb{P}_\Pi$,

iii. add the constraint '$\leftarrow \widetilde{p}\ ,\ p$', which is also a general program rule, into $\widetilde{\Pi}$.

Similarly, a set of literals $S \subseteq Lit$ can be turned into a set of atoms $\widetilde{S} \subseteq \mathbb{P}$ by following item (i). The following lemma describes this syntactic transformation.

Lemma 1.1 *Given an extended logic program* Π *and a consistent set* $S \subset Lit$,

$$S \textit{ is an answer set of } \Pi \quad \textit{iff} \quad \widetilde{S} \textit{ is a stable model of } \widetilde{\Pi} \cup Cons.$$

One can refer Gelfond and Lifschitz [1991] for the proof. This lemma is generalised in Subsection 1.3.3 to arbitrary logic programs.

The answer set semantics is not *contrapositive* (with respect to '$\leftarrow$' and '$\sim$') in the sense that it distinguishes, for example, between the rules '$q \leftarrow p$' and '$\sim p \leftarrow \sim q$'. Thus, we see that we cannot interpret $\leftarrow$ as material implication. The first example below discusses this fact.

E.1 While the answer set of the program $\Pi_1 = \left\{ \begin{array}{lll} q & \leftarrow & \\ p & \leftarrow & q \end{array} \right.$ is $\{p, q\}$, the answer sets of the programs $\Pi_2 = \left\{ \begin{array}{lll} q & \leftarrow & \\ \sim q & \leftarrow & \sim p \end{array} \right.$ and $\Pi_3 = \left\{ \begin{array}{lll} \sim p & \leftarrow & \\ \sim q & \leftarrow & \sim p \end{array} \right.$ are respectively $\{q\}$ and $\{\sim p, \sim q\}$. Similarly, the programs $\Pi_4 = \left\{ \begin{array}{lll} \sim p & \leftarrow & \\ p & \leftarrow & \sim q \end{array} \right.$ and $\Pi_5 = \left\{ \begin{array}{lll} \sim p & \leftarrow & \\ q & \leftarrow & \sim p \end{array} \right.$ have respectively the answer sets $\{\sim p\}$ and $\{\sim p, q\}$.

E.2 The closed world assumption (CWA) for p and $\sim p$ can be formalised respectively by the following extended logic program rules:

$$\sim p \leftarrow not\ p \ \text{ and } \ p \leftarrow not\ \sim p.$$

The program Π_6 uniquely containing the former and the program Π_7 containing just the latter have exactly one answer set, i.e., respectively $\{\sim p\}$ and $\{p\}$. Consider now another one-rule logic program Π_8:

$$\sim q \leftarrow not\ p.$$

The only answer set of Π_8 is $\{\sim q\}$. So, for rules of this kind we simply obtain the answer set including only the literal in the head except for the cases $p \leftarrow not\, p$ and $\sim p \leftarrow not\, \sim p$ where the rules have no answer sets at all.

E.3 The program $\Pi_9 = \left\{ \begin{array}{lll} r & \leftarrow & \\ p & \leftarrow & not\ q \\ q & \leftarrow & not\ p \\ \sim r & \leftarrow & not\ p \end{array} \right.$ has exactly two answer sets: $\{p, r\}$ and $\{q, r, \sim r\}$. However, the latter turns out to be *Lit* since it contains two complementary literals. However, according to today's widely accepted semantics, the latter is not an answer set anymore, and so Π_9 has just one answer set $\{p, r\}$.

E.4 Finally, the extended program $\Pi_{11} = \left\{ \begin{array}{lll} q & \leftarrow & p \\ \sim q & \leftarrow & p \\ p & \leftarrow & not\ \sim p \end{array} \right.$ has no answer sets.

Transforming Π_{11} into a corresponding general program $\widetilde{\Pi}_{11} = \left\{ \begin{array}{lll} q & \leftarrow & p \\ \widetilde{q} & \leftarrow & p \\ p & \leftarrow & not\ \widetilde{p} \end{array} \right.$

we see that the latter has the following stable model: $\{p, q, \widetilde{q}\}$. This example thus shows that the consistency constraint in Lemma 1.1 is indeed essential.

As in general **LP**, an extended logic program may also have finitely many answer sets, including none. However, a 'well-behaved' extended program has exactly one answer set, and this set should be consistent, i.e., different from *Lit*.

Extended logic programs are useful for representing incomplete information: they can include explicit negative information via strongly negated propositional variables (negative literals). In other words, in the language of extended **LP**, we can distinguish between a query l which fails in the sense that it *does not succeed* (i.e., *not* l holds) and a query which fails in the stronger sense that its *negation succeeds* (i.e., $\bar{l}$ holds where $\bar{l}$ is the complementary literal of l, see Subsection 1.1.1, second paragraph for a detailed explanation of $\bar{l}$). A consistent extended program Π is supposed to return an answer *yes*, *no* or *unknown* for a ground query "l ?", depending on whether its answer sets contain l, $\bar{l}$, or neither. To express it more clearly, Π's answer to a literal query "l?" is *yes* if l is contained in all its answer sets, *no* if $\bar{l}$ is included in all its answer sets and *unknown* otherwise (Baral and Gelfond [1994]). However, there is one exception: when Π is contradictory it answers both *yes* and *no* to all literal queries because the unique answer set of a contradictory program is *Lit*.

Although general logic programs are syntactically a special case of extended logic programs, one should note that a general logic program has different semantic interpretations in general **LP** and extended **LP**: while the absence of an atom p in a stable model of a general program indicates that p is false, so the program answers *no* to the query "p?"; the same program, when viewed as an extended program, interprets the absence of the same atom in the corresponding answer set as "p is unknown". Expectedly, such semantic equivalence is just obtained when an extended rule representing CWA is added into the program for every $p \in \mathbb{P}_\Pi$. In fact, such addition is sufficient for all atoms $p \in \mathbb{P}_{head(r)}$ and every $r \in \Pi$.

Note that we consider answer sets in this work both as 'incomplete theories' following Gelfond and Lifschitz's approach and also as '3-valued models' following the approach described in, for instance, (Fitting [1985]; Przymusinski [1989]; Van Gelder et al. [1991]). In this sense, stable models are not incomplete, in other words, they are two-valued. However, for a program having more than one answer set (or stable model) points out incompleteness of the program in another sense: several different interpretations of the world to which the program refers.

Reexamining the examples above, all programs in the first two examples (see Example E.1 and Example E.2) behave well in the sense that they have just one consistent answer set. The rule of Π_8 intuitively means: "q is false if there is no evidence that p is true". The program Π_8 respectively answers to the queries "p ?" and "q?", *unknown* and *false* since its answer set $\{\sim q\}$ does not include the former,

yet does include the strong negation of the latter. The program Π_9 is consistent since it has one consistent answer set, and responds *yes* the queries "p?" and "r?", but *unknown* to the rest of literal queries. Finally, the program Π_{11} is inconsistent since it has no answer sets.

The following part discusses a further extension of extended **LP** by a new operator called epistemic disjunction, which is necessary to represent disjunctive information about the world.

Disjunctive logic programs

A program composed of a finite set of rules $\bigvee_{1}^{m} \varphi_i \leftarrow \left(\bigwedge_{m+1}^{n} \varphi_i \right) \wedge \left(\bigwedge_{n+1}^{k} not\, \varphi_i \right)$ (see Definition 1.3) is called a *disjunctive logic program* for $0 \leq m \leq n \leq k$ and when all φ_i's are literals. Therefore, a disjunctive program (Gelfond and Lifschitz [1991]) is a finite collection of rules of the following explicit form (see Definition 1.2)

$$l_1 \, or \, \ldots \, or \, l_m \leftarrow (l_{m+1}, \ldots, l_n), (not \, l_{n+1}, \ldots, not \, l_k) \tag{1.7}$$

where $l_i \in Lit$ for every $i = 1, \ldots, k$. Briefly, bodies are as before (see Definition 1.6), but heads allow for disjunctions of literals. The term *disjunctive database* is sometimes used in the same meaning with a disjunctive logic program.

In our context, extended logic programs are a special case of disjunctive logic programs. However, some authors prefer to make such a separation: they call general logic programs enriched by the epistemic disjunction operator disjunctive logic programs and programs that further contain strong negation extended disjunctive programs.

Epistemic disjunction (*or*) is nonclassical, so it is different from both classical disjunction ($\vee$) and exclusive disjunction ($\oplus$). The last is expressed by means of classical disjunction as follows: $\varphi \oplus \psi = (\varphi \vee \psi) \wedge -(\varphi \wedge \psi)$ where '$-$' refers to classical negation. As to epistemic disjunction, one can represent '$\varphi \; or \; \psi \leftarrow$' in the language of general **LP** by the following two rules: '$\varphi \leftarrow \; not \; \psi$' and '$\psi \leftarrow \; not \; \varphi$'. At first sight the reader may feel a possible similarity between *or* and $\oplus$, but we will soon show semantically that they are in fact different (see Example D.4). One can also define the formula '$\varphi \; or \; \psi$' as a shortcut for '$\sim(\sim\varphi, \sim\psi)$', but we have not discussed such a generalised language in this work in which strong negation precedes an arbitrary formula. We have also preferred to denote epistemic disjunction by '*or*' except compact representations (see Definition 1.3). However, in the literature, the more common notation for this connective is ';', and even one can come across '$\vee$'. The latter notation stresses the fact that the rule '$\varphi \; or \; \psi$' can be translated to a propositional formula '$p \vee q$', but certainly not in classical logic (see Example D.3). We will return to this point in Section 3.1.2.

The meaning of '*or*' is given by the semantics of disjunctive databases. A formula '$\varphi \; or \; \psi$' is interpreted epistemically and means "φ is believed to be true

or ψ is believed to be true". This is why we call this disjunction specifically *epistemic*. On the other hand, the formulas '$\varphi \vee \psi$' and '$\varphi \oplus \psi$' respectively read "φ is true or ψ is true" and "either φ or ψ is true, but not both".

As to the semantics of disjunctive **LP**, we mainly follow the slight generalisation of the answer set definition given in the previous section, but the reduct definition is the same as before. However, when a program does not contain any negative literals, we can also follow the stable model definition. In this case, note that a reduct Π^S of a disjunctive program Π (w.r.t. S) does not have a least Herbrand model anymore: it just has minimal ones, if any.

Given a disjunctive program Π and a set $S \subseteq \mathit{Lit}$, the rules of the reduct Π^S, w.r.t. S, are in the following explicit form:

$$l_1 \mathit{\ or\ } \ldots \mathit{\ or\ } l_m \leftarrow l_{m+1}\,, \ldots\,, l_n \qquad \text{for } 0 \leq m \leq n.$$

Then, minimal models of Π^S are the models $X \subseteq \mathit{Lit}$ satisfying:

- for every $r \in \Pi^S$, if $l_i \in X$ for every $m+1 \leq i \leq n$, then $l_j \in X$ for some j such that $1 \leq j \leq m$, and
- if X contains a pair of complementary literals, then $X = \mathit{Lit}$ (as before).

Finally, if S is one of such X's then we say S is answer set of Π. Here are some examples: The first example refers to the nonexistence of a least Herbrand model, instead the existence of minimal models for the reduct.

D.1 The disjunctive program

$$\Pi_1 = \left\{ \begin{array}{rcl} p \mathit{\ or\ } q & \leftarrow & r\,, \mathit{not\ } s \\ r & \leftarrow & \mathit{not\ } q \end{array} \right.$$

has a unique stable model: $\{p, r\}$. (Note that this program does not contain any negative literals.) To see this, we first take the reduct

$$\Pi_1^{\{p,r\}} = \left\{ \begin{array}{rcl} p \mathit{\ or\ } q & \leftarrow & r \\ r & \leftarrow & . \end{array} \right.$$

The minimal models of $\Pi_1^{\{p,r\}}$ are $\{p, r\}$ and $\{q, r\}$. Hence, $\{p, r\}$ is a stable model of Π_1 since it satisfies the fixed point property. On the other hand, through a similar reduct discussion, we see that $\{q, r\}$ is not a stable model of this program.

D.2 The extended program $\Pi_2 = \left\{ \begin{array}{rcl} p \mathit{\ or\ } {\sim}p & \leftarrow & \mathit{not\ } p \\ & \leftarrow & {\sim}p \end{array} \right.$ has no answer sets.

We have somewhat mentioned above the difference of *or* from classical disjunction $\vee$ and exclusive disjunction $\oplus$. The next example supports semantically the difference between *or* and $\vee$.

D.3 We first consider a simple program $\Pi_3 = \left\{ \begin{array}{rcl} q & \leftarrow & p \end{array} \right.$ which has a unique answer set $\emptyset$. Appending the disjunctive rule

$$p \; or \; {\sim}p \leftarrow \tag{1.8}$$

to Π_3, we get a new program $\Pi_4 = \left\{ \begin{array}{rcl} q & \leftarrow & p \\ p \, or \, {\sim}p & \leftarrow & \end{array} \right.$ whose answer sets are exactly $\{p, q\}$ and $\{{\sim}p\}$. Thus, we see that the inclusion of rule (1.8) makes a striking change, in contrast to classical logic, in which the law of excluded middle is valid.

The following example specifies the difference between all three connectives: *or*, $\vee$ and $\oplus$.

D.4 The disjunctive program $\Pi_5 = \{p \; or \; q \leftarrow\}$ has exactly two answer sets: $\{p\}$ and $\{q\}$ (see Example G.2 and note that the answer sets of these programs are the same). However, adding the rule '$p \leftarrow$' into Π_5 slightly changes its answer sets. To express it more clearly, $\Pi_6 = \left\{ \begin{array}{rcl} p \; or \; q & \leftarrow & \\ p & \leftarrow & \end{array} \right.$ has a unique answer set: $\{p\}$. Moreover, making one step forward, i.e., having '$q \leftarrow$' added into Π_6, $\Pi_7 = \left\{ \begin{array}{rcl} p \; or \; q & \leftarrow & \\ p & \leftarrow & \\ q & \leftarrow & \end{array} \right.$ has the following unique answer set: $\{p, q\}$. As a result, in intuitive means, while Π_6 is interpreted in the same way for all three connectives, Π_5 makes *or* differ from $\vee$, and Π_7 makes it differ from $\oplus$.

To close with the complexity, we say that finding a stable model of a disjunctive logic program (but not extended) is slightly more complex than finding a stable model of a general logic program: it is Σ_2^P-complete (Eiter and Gottlob [1993]).

Negation as failure in the head of logic programs

A disjunctive logic program that allows for the NAF operator in the head as a negative conclusion is known as a *generalised disjunctive program* (GDP, for short) and has the following form:

$$(l_1 \; or \; \ldots \; or \, l_x) \; or \; (not \, l_{x+1} \; or \; \ldots \; or \, not \, l_m) \leftarrow (l_{m+1}, \ldots, l_n), \, (not \, l_{n+1}, \ldots, not \, l_k)$$

where $0 \leq x \leq m \leq n \leq k$ and $l_i \in Lit$ for every $i = 1, \ldots, k$ (Inoue and Sakama [1998]; Lifschitz [1996]). They were first introduced by Lifschitz and Woo (Lifschitz and Woo [1992]) as a subset of the **MB-NF** logic (see Appendix C.2).

The answer set semantics we have followed so far satisfies the antichain property: the answer set of a program cannot be a proper subset of another answer set of the same program. The semantics proposed for GDPs is a slight generalisation of the usual answer set semantics. The main difference of the latter is that the principle of minimality (i.e., antichain property) does not necessarily hold since NAF is now allowed to appear in the head. So, GDPs are useful for representing knowledge in various domains, such as abductive logic programming, in which the minimality condition is too strong.

We again follow a similar reduct definition as followed in the second convention (see Definition R2.1 and Definition R2.2) to eliminate NAF, also for ones occurring in the head. However, unlike the definitions given before, we do not allow here for an inconsistent answer set *Lit*. For instance, a simple GDP $\Pi_1 = \{not\, p \leftarrow\}$ has one answer set $\emptyset$, but $\Pi_2 = \left\{ \begin{array}{lll} p & \leftarrow & \\ & \leftarrow & p \end{array} \right.$ has no answer sets in the new context. Note that Π_2, when viewed as a disjunctive or an extended program, *Lit* appears to be an answer set of this program. Another GDP $\Pi_3 = \{p\, or\, not\, p \leftarrow\}$ has two nonminimal answer sets: $\emptyset$ and $\{p\}$. Finally, it is interesting to observe that such rules make an incoherent (or inconsistent) program coherent: for instance, as a GDP $\Pi_4 = \left\{ \begin{array}{lll} p & \leftarrow & not\, p \\ \sim q & \leftarrow & \end{array} \right.$ has no answer sets. Note that as a disjunctive program the unique answer set of Π_4 is *Lit*. However, $\Pi_3 \cup \Pi_4$ has a (unique) consistent answer set $\{\sim q, p\}$, so the resulting program turns out to be coherent.

In *acyclic* programs (Apt and Bezem [1991]), NAF in the head can be shifted to the body without changing the answer sets of the program. For example, replacing a rule r of the form

$$not\, l_1\, or\, \ldots\, or\, not\, l_k \leftarrow body(r)$$

in any program by the constraint

$$\bot \leftarrow l_1\, , \ldots\, , l_k\, , body(r) \tag{1.9}$$

does not affect the answer sets of the program since they are strongly equivalent (Lifschitz et al. [2001]). This concept is discussed in Subsection 1.1.4 in detail. This fact is further interesting because the role of constraints in a logic program is well understood: adding a constraint to a program eliminates its answer sets that violate the constraint (see Subsection 1.1.3.1 for part of the discussion). However, it is not always possible to easily eliminate *not* in the head. Note that the program Π_3 above cannot be transformed to $\Pi_3' = \{p \leftarrow p\}$ which has a unique answer set,

i.e., $\emptyset$. We will return to this example in Subsection 1.2.3. However, Π_1 and Π_2 can be safely transformed to $\Pi_1' = \{\leftarrow p\}$ and $\Pi_2' = \left\{ \begin{array}{rl} p & \leftarrow \\ not\, p & \leftarrow \end{array} \right.$.

To close with the computational complexity, it remains the same as with the complexity of nonextended disjunctive programs: deciding the existence of an answer set of a GDP Π is Σ_2^P-complete (Inoue and Sakama [1998]).

Nested expressions in logic programs

Nested expressions are formed from literals using NAF, conjunction, epistemic disjunction and even *if-then-else* constructs that can be nested arbitrarily. The latter (conditional expressions) is given in the form of '$\varphi \rightarrow \psi\, or\, \chi$' and is an abbreviation for the formula '$(\varphi\,,\, \psi)\, or\, (not\, \varphi\,,\, \chi)$'. For instance, the rule

$$s \leftarrow (p \rightarrow q\, or\, r), t$$

declaratively has the same meaning as the set of rules

$$\begin{array}{rcl} s & \leftarrow & p\,,\, q\,,\, t \\ s & \leftarrow & not\, p\,,\, r\,,\, t \end{array}$$

yet, the former is preferable since it is more concise. We call rules with nested expressions *nested rules* and finite sets of such rules *nested programs*.

The ASP programs we have discussed so far only allow a list of syntactically simple expressions in its program rules, but the syntax of Prolog also permits more complex nested expressions. So, nested programs help us catch a 1-1 correspondence between them. However, the study of equivalence transformations of nested programs shows that every nested rule is equivalent, in the stronger sense, to a set of disjunctive rules, possibly with NAF in the heads. This equivalence means more than just having the same answer sets and is formally defined as follows: φ is equivalent to ψ (symbolically, $\varphi \Leftrightarrow \psi$) if for every consistent $S_1, S_2 \subset Lit$, "$S_1 \models \varphi^{S_2}$ if and only if $S_2 \models \psi^{S_1}$". We will see this equivalence as *strong equivalence* in Subsection 1.1.4. As a result, nested programs allow us to express ASP programs in a more compact way.

Here is a list of useful equivalences of formulas (Lifschitz et al. [1999]). Note that we can consider any formula in $\mathcal{L}_{\mathbf{ASP}}$ as a 'nested fact'. For every formulas φ, ψ and χ in $\mathcal{L}_{\mathbf{ASP}}$, we have:

i) commutativity and associativity hold for both , and *or*.

ii) $\varphi\,,\, (\psi\, or\, \chi) \Leftrightarrow (\varphi\,,\, \psi)\, or\, (\varphi\,,\, \chi)$ and vice versa.

iii) De Morgan's laws hold for both , and *or*, i.e., $not(\varphi\,,\, \psi) \Leftrightarrow not\, \varphi\, or\, not\, \psi$ and $not(\varphi\, or\, \psi) \Leftrightarrow not\, \varphi\,,\, not\, \psi$.

iv) $not\, not\, not\, \varphi \Leftrightarrow not\, \varphi$

v) conjunction and disjunction with $\top$ or $\bot$ are as usual.

vi) $not\, \top \Leftrightarrow \bot$ and $not\, \bot \Leftrightarrow \top$.

vii) $p\,, \sim p \Leftrightarrow \bot$ and hence by item (vi) $not\, p\, or\, not \sim p \Leftrightarrow \top$.

The following collection describes further equivalence transformations of nested rules.

I) $\varphi\,, \psi \leftarrow \chi$ is equivalent to $\left\{ \begin{array}{lcl} \varphi & \leftarrow & \chi \\ \psi & \leftarrow & \chi \end{array} \right.$.

II) $\varphi \leftarrow \psi\, or\, \chi$ is equivalent to $\left\{ \begin{array}{lcl} \varphi & \leftarrow & \psi \\ \varphi & \leftarrow & \chi \end{array} \right.$.

III) $\varphi \leftarrow \psi\,, not\, not\, \chi$ is equivalent to $\varphi\, or\, not\, \chi \leftarrow \psi$.

IV) $\varphi\, or\, not\, not\, \psi \leftarrow \chi$ is equivalent to $\varphi \leftarrow not\, \psi\,, \chi$.

The semantics of nested programs, beside other approaches, is based on answer set semantics which is similar to the one given in the previous subsection. We still find it useful to include a generalised reduct definition. The reduct of a formula and a rule relative to a consistent set $X \subset Lit$ is recursively defined as follows (Lifschitz et al. [1999]):

R3.1 for every literal $l \in Lit$, $l^X = l$.

R3.2 $(\varphi \otimes \psi)^X = \varphi^X \otimes \psi^X$ where $\otimes$ stands for , and or.

R3.3 $(not\, \varphi)^X = \left\{ \begin{array}{ll} \bot & \text{if } X \models \varphi^X \\ \top & \text{otherwise.} \end{array} \right.$

R3.4 $(\varphi \leftarrow \psi)^X = \varphi^X \leftarrow \psi^X$.

Here are some examples.

N.1 Using item (v) above, we conclude that $\Pi_0 = \{p\, or\, not\, p \leftarrow\}$ has two answer sets $\emptyset$ and $\{p\}$ (see previous subsection), and $\Pi_0' = \{p\, or\, not\, q \leftarrow\}$ has just one, namely $\emptyset$. However, adding the rule $q\, or\, not\, p \leftarrow$ to both programs results in the same answer sets of the resulting programs: $\emptyset$ and $\{p, q\}$.

N.2 The nested program $\Pi_1 = \{q \leftarrow p \mathit{\ or\ } {\sim}p\}$ has a unique answer set $\emptyset$. Note that Π_1 is equivalent to the extended program $\Pi_1' = \left\{ \begin{array}{lcl} q & \leftarrow & p \\ q & \leftarrow & {\sim}p \end{array} \right.$ (by item (II) above). It is now easier to also see that adding the rule $p \leftarrow$ into Π_1 results in the unique answer set $\{p, q\}$ of the new program. As a result of this example, we conclude that the formula $p \mathit{\ or\ } {\sim}p$ is not equivalent to $\top$.

N.3 Another one-rule nested program $\Pi_2 = \{p \leftarrow (q \rightarrow r \mathit{\ or\ not\ } s)\}$ has a unique answer set $\{p\}$: to show this, we first turn Π_2 into its explicit form $\Pi_2' = \{p \leftarrow (q\,,\, r) \mathit{\ or\ } (\mathit{not\ } q\,,\, \mathit{not\ } s)\}$, then find the reduct ${\Pi_2'}^{\{p\}} = \{p \leftarrow (q\,,\, r) \mathit{\ or\ } (T\,,\, T)\}$ and finally see that $\{p\}$ satisfies the fixed point property. To produce the reduct, we could alternatively work with the general program $\Pi_2'' = \left\{ \begin{array}{lcl} p & \leftarrow & q\,,\, r \\ p & \leftarrow & \mathit{not\ } q\,,\, \mathit{not\ } s \end{array} \right.$ which is equivalent to Π_2' (by item (II)), and see ${\Pi_2''}^{\{p\}} = \left\{ \begin{array}{lcl} p & \leftarrow & q\,,\, r \\ p & \leftarrow & \end{array} \right.$. The rest is obvious.

N.4 The nested program $\Pi_3 = \{{\sim}q\,,\, \mathit{not\ } p \leftarrow \mathit{not\ } r\}$ has a unique answer set $\{{\sim}q\}$. Note that Π_3 is equivalent to $\Pi_3' = \left\{ \begin{array}{rcl} {\sim}q & \leftarrow & \mathit{not\ } r \\ \mathit{not\ } p & \leftarrow & \mathit{not\ } r \end{array} \right.$ (by item (I)), and even to the extended program $\Pi_3'' = \left\{ \begin{array}{lcl} {\sim}q & \leftarrow & \mathit{not\ } r \\ & \leftarrow & p\,, \mathit{not\ } r \end{array} \right.$. The rest is obvious. At this point, it is useful to notice that $\mathit{not\ } p \leftarrow \mathit{not\ } r$ is not equivalent to $r \leftarrow p$ (in the stronger sense). More generally, we say that the answer set semantics is not contrapositive w.r.t. $\leftarrow$ and *not*.

Finally, we discover an interesting fact by means of nested programs under answer set semantics: double *not* does not cancel out in general. Here is an example.

N.5 The program $\Pi_4 = \{p \leftarrow \mathit{not\ not\ } p\}$ has two answer sets $\emptyset$ and $\{p\}$: Π_4 is equivalent to $\Pi_4' = \{p \leftarrow \top\,,\, \mathit{not\ not\ } p\}$ (by item (v)) and then also to $\Pi_4'' = \{p \mathit{\ or\ not\ } p \leftarrow \top\}$ (by item (III)). However, $\Pi_5 = \{p \leftarrow p\}$ has just one answer set, i.e., $\emptyset$.

Note that when *not* is nested, i.e., applied to formulas containing *not*, we should be further careful: we first apply the usual reduction procedure starting from the innermost *not*, so we get $(\mathit{not\ } p)^{\{p\}} = \bot$. Then, since $\{p\} \not\models \bot$ (or, using $\mathit{not\ } \bot \Leftrightarrow \top$ in item (vi)), we have $(\mathit{not\ not\ } p)^{\{p\}} = \top$ and hence $(p \leftarrow \mathit{not\ not\ } p)^{\{p\}} = p \leftarrow$. The rest is obvious.

1.1.3 Other language extensions: new constructs in ASP

It has been observed in time that general **LP** with its stable model semantics is not adequate for many interesting domains. They lack expressivity to represent, for instance, choices over subsets as well as cardinality and weight constraints. Therefore, some new types of rules (constructs) were introduced in order to further enhance the expressivity of the **ASP** language. Such constructs also allow us to write more concise ASP programs in the sense that they mostly abbreviate a set of general rules in a very compact representation. They share the language of general **LP**, and accordingly, their semantics extend the stable model semantics of general logic programs. Such constructs can be embedded into general **LP**, but usually with a cost of increase in the rule number.

1.1.3.1 Integrity constraints

An *integrity constraint* r is an expression of the form

$$\leftarrow p_1 , \ldots , p_m , not\, p_{m+1} , \ldots , not\, p_n \tag{1.10}$$

where $1 \leq m \leq n$ and $p_i \in \mathbb{P}$ for $1 \leq i \leq n$. So, it is a special case of general logic programs in which $headr(r) = \bot$ (see Subsection 1.1.1). However, an integrity constraint 1.10 can also be translated into a general program rule as follows:

$$q \leftarrow p_1 , \ldots , p_m , not\, p_{m+1} , \ldots , not\, p_n , not\, q$$

where $q \in \mathbb{P}$ is an atom, not appearring in the constraint 1.10, i.e., $q \in \mathbb{P} \setminus \mathbb{P}_{body(r)}$. An integrity constraint in the form of 1.10 can also appear in another disguise:

$$not\, p_1 , \ldots , not\, p_x \leftarrow p_{x+1} , \ldots , p_m , not\, p_{m+1} , \ldots , not\, p_n .$$

This fact is clear by equivalence transformations of rules (see 1.9) and note that we still do not infer a literal directly. In this sense, they are also a special case of generalised disjunctive rules.

Using integrity constraints extended by strong negation, we can formalise the *coherence principle*: '$not\, p \leftarrow {\sim}p$' and '$not\, {\sim}p \leftarrow p$', which is nothing but '$\leftarrow p , {\sim}p$'. We have already mentioned such constraints implicitly as conjuncts of $Cons$ while eliminating strong negation in an extended logic program, resulting in a general logic program. Coherence principle is an important property for programs including both NAF and classical negation. Note that these schemas for the coherence principle are converse to the rules for *closed world assumption* (CWA) (see Example E.2).

The idea underlying this form of constructs is to eliminate unwanted solution candidates. Here is a simple example: while the program $\Pi_1 = \{p \, or \, q \leftarrow\}$ has two

stable models $\{p\}$ and $\{q\}$, the programs $\Pi_1 \cup \{\leftarrow p\}$ and $\Pi_1 \cup \{\leftarrow \mathit{not}\, p\}$ have respectively the unique stable models $\{q\}$ and $\{p\}$. The following example gives the idea better. We have seen in Example E.3 that the ASP program

$$\Pi = \left\{ \begin{array}{rcl} r & \leftarrow & \\ p \,\mathit{or}\, q & \leftarrow & \\ \sim r & \leftarrow & \mathit{not}\, p \end{array} \right.$$

has two stable models $\{p, r\}$ and *Lit*. Hence, adding the integrity constraint '$\leftarrow q$' into Π will simply eliminate the latter which is indeed an undesired solution.

1.1.3.2 Choice rules

A *choice rule* r is an expression of the form

$$\{p_1, \ldots, p_m\} \leftarrow p_{m+1}\,, \ldots\,, p_n\,, \mathit{not}\, p_{n+1}\,, \ldots\,, \mathit{not}\, p_k \tag{1.11}$$

where $1 \leq m \leq n \leq k$ and $p_i \in \mathbb{P}$ for $1 \leq i \leq k$. Different from other program rules we have seen so far, now the head of the rule is given as a set of propositional variables, i.e., $\mathit{head}(r) = \{p_1, \ldots, p_m\} \subset \mathbb{P}$, and it says: "choose arbitrarily which of the atoms $p_1, \ldots, p_m$ (including none) to add into the stable model". Therefore, in this form of constructs, a choice is made over $2^{\mathit{head}(r)}$. For example, an ASP program containing only the choice rule '$\{p, q, r\} \leftarrow$' has 8 stable models: any subset of $\{p, q, r\}$ is a stable model. Note that we generalise the stable model semantics for choice rule structures, and in contrast to stable model semantics defined in general **LP**, the new semantics is not based on a minimality condition relative to set inclusion.

Intuitively, choice rules work as follows: if $\mathit{body}(r)$ of a choice rule r is satisfied by a possible stable model X at hand, then the rule r motivates X to include any number (including none) of atoms from the set $\mathit{head}(r)$. Such rules are used when one wants to implement optional choices. For instance, an ASP program

$$\Pi = \left\{ \begin{array}{rcl} q & \leftarrow & \\ \{r, s\} & \leftarrow & \mathit{not}\, p \end{array} \right.$$

has four stable models: $\{q\}$, $\{q, s\}$, $\{q, r\}$ and $\{q, r, s\}$. Further adding the integrity constraint '$\leftarrow \mathit{not}\, r$' into Π eliminates the first two stable models, and the resulting program Π' has just two stable models: $\{q, r\}$ and $\{q, r, s\}$.

Remember how we have translated program rules into propositional formulas. Similarly, we can also treat, for example, a choice rule '$\{p, q, r\} \leftarrow$' as a conjunction of three *excluded middle* formulas:

$$(p \vee \neg p) \wedge (q \vee \neg q) \wedge (r \vee \neg r).$$

We can easily generalise this transformation to an arbitrary choice rule structure.

1.1.3.3 Cardinality rules

A *cardinality rule* r an expression of the form

$$p_0 \leftarrow l\ \{p_1, \ldots, p_m, not\, p_{m+1}, \ldots, not\, p_n\} \tag{1.12}$$

where $p_i \in \mathbb{P}$ for $0 \leq i \leq n$ such that $0 \leq m \leq n$ and l is a nonnegative integer. In this form of constructs, the body formula

$$p_1\ ,\ \ldots\ ,\ p_m\ ,\ not\, p_{m+1}\ ,\ \ldots\ ,\ not\, p_n$$

is given in a set notation in which any element is a conjunct of this formula. This set, together with a further restrictive condition l, form $body(r)$. So, the sign ',' appearing in such rules is the usual comma, but not the conjunction in $\mathcal{L}_{ASP}$. The body of this construct is referred to as a *cardinality constraint*, and it allows us to put a restriction on the body formula that allows for the cardinality increase of already existing stable models (see Example C.1). This restriction is made by a number l which acts as a lower bound on $body(r)$. However, when $l = 0$ a cardinality rule 1.12 behaves as a fact '$p_0 \leftarrow$'.

Informally, such rules work as follows: if a stable model $X \subset \mathbb{P}$ satisfies at least l elements of the conjuncts making up $body(r)$, then p_0 belongs to the stable model. So, in this context, X does not necessarily satisfy $body(r)$ as a whole for $head(r)$ to be included in the stable model, instead it should satisfy a subformula of it containing at least l conjuncts (see Example C.1 and Example C.2). Formally, this condition is given as:

$$l \leq |(\{p_1, \ldots, p_m\} \cap X) \cup (\{p_{m+1}, \ldots, p_n\} \setminus X)|\,.$$

The idea underlying such rules is to lower the cardinality increase of stable models (see Example C.3).

General program rules (see Definition 1.7) are special forms of such constructs: a rule '$r = p_0 \leftarrow p_1\ ,\ \ldots\ ,\ p_m\ ,\ not\, p_{m+1}\ ,\ \ldots\ ,\ not\, p_n$' can be represented by the cardinality rule '$p_0 \leftarrow l\ \{p_1, \ldots, p_m, not\, p_{m+1}, \ldots, not\, p_n\}$' where the lower bound $l = \left|\mathbb{P}_{body(r)}\right|$ when p_i's are all different for every $i = 1, \ldots, n$.

A *cardinality rule with an upper bound* has the following explicit form

$$p_0 \leftarrow l\ \{p_1, \ldots, p_m, not\, p_{m+1}, \ldots, not\, p_n\}\ u \tag{1.13}$$

where $p_i \in \mathbb{P}$ for $0 \leq i \leq n$ such that $0 \leq m \leq n$ and $0 \leq l \leq u$. The cardinality constraint of such rules contains also an upper bound u in addition to a lower bound l, and a stable model $X \subset \mathbb{P}$ satisfies it if X satisfies x number of conjuncts of $body(r)$ such that $l \leq x \leq u$. We can formalise this condition as follows:

$$l \leq |(\{p_1, \ldots, p_m\} \cap X) \cup (\{p_{m+1}, \ldots, p_n\} \setminus X)| \leq u.$$

A cardinality constraint structure can also appear in the head of a rule in some form of constructs, and it determines the range of the number of atoms to be or not be added into X when $body(r)$ is satisfied by a stable model X at hand. Now, we see some examples.

C.1 Recall that the rule '$q \leftarrow$' has a unique stable model $\{q\}$, but together with the following cardinality rule

$$p \leftarrow 1\ \{q, r\} \tag{1.14}$$

the resulting ASP program $\Pi_1 = \left\{ \begin{array}{lll} q & \leftarrow & \\ p & \leftarrow & 1\ \{q, r\} \end{array} \right.$ has a unique stable model $\{p, q\}$. However, when we replace the lower bound $l = 1$ by 2, the resulting rule has no effect on the previous model $\{q\}$. Note that in this example the lower bounds 1 and 0 makes the same effect on the stable model. However, this is not a striking example and does not help any further than introducing the structure.

C.2 Recall that $q \leftarrow not\ p$ has exactly one stable model $\{q\}$ (see Example G.1). When we append the cardinality rule 1.14 of the previous example to this rule, we see that the resulting ASP program

$$\Pi_2 = \left\{ \begin{array}{lll} q & \leftarrow & not\, p \\ p & \leftarrow & 1\ \{q, r\} \end{array} \right.$$

has a unique stable model $\{q, p\}$. Note that $\{q, p\}$ is a model of the latter rule although it is not minimal. On the other hand, the ASP program

$$\Pi_3 = \left\{ \begin{array}{lll} & \leftarrow & r\ ,\ not\, p \\ r & \leftarrow & 1\ \{p, not\, q\} \end{array} \right.$$

has no stable models. Note that the first rule in the program Π_3 is an integrity constraint, and has only one stable model, i.e., the empty set. Therefore, our stable model at hand is $\emptyset$. When we consider the effect of the latter rule, which is a cardinality constraint, to $\emptyset$, we should add by definition r into this stable model, resulting in $\{r\}$. However, $\{r\}$ violates the first rule, i.e., does not satisfy it anymore.

C.3 Recall that the ASP program

$$\Pi' = \left\{ \begin{array}{rcl} q & \leftarrow & \\ & \leftarrow & not\, r \\ \{r, s\} & \leftarrow & not\, p \end{array} \right.$$

in Subsection 1.1.3.2 has two stable models: $S_1 = \{q, r\}$ and $S_2 = \{q, r, s\}$. The effect of the cardinality rule '$p \leftarrow 0\ \{q, s, not\, r\}\ 1$' on these models will be as follows: $S_1' = S_1 \cup \{p\} = \{q, r, p\}$ and $S_2' = S_2 = \{q, r, s\}$ which are the stable models of the resulting program

$$\Pi'' = \left\{ \begin{array}{rcl} q & \leftarrow & \\ & \leftarrow & not\, r \\ \{r, s\} & \leftarrow & not\, p \\ p & \leftarrow & 0\ \{q, s, not\, r\}\ 1. \end{array} \right.$$

1.1.3.4 Weight rules

A *weight rule* is of the form

$$p_0 \leftarrow l\ \{p_1 = w_1\, ,\ \ldots\ ,\ p_m = w_m\, ,\ not\ p_{m+1} = w_{m+1}\, ,\ \ldots\ ,\ not\ p_n = w_n\}$$

where $0 \leq m \leq n$, $p_i \in \mathbb{P}$ for $0 \leq i \leq n$ and w_i's are integers. A weighted form $p_i = w_i$ (or, $not\ p_i = w_i$) associates p_i (or, $not\ p_i$) with a weight w_i. Note that a cardinality rule is a weight rule where $w_i = 1$ for $1 \leq i \leq n$.

A *weight constraint with an upper bound* is of the form

$$l\ \{p_1 = w_1\, ,\ \ldots\ ,\ p_m = w_m\, ,\ not\, p_{m+1} = w_{m+1}\, ,\ \ldots,\ an = wn\}\ u$$

where $1 \leq m \leq n$, $p_i \in \mathbb{P}$, and l, u and w_i are integers for $1 \leq i \leq n$. A weight constraint is satisfied by a stable model X, if

$$l \leq \Big(\sum_{1 \leq i \leq m\ ,\ p_i \in X} w_i \ + \sum_{m < i \leq n\ ,\ p_i \notin X} w_i \Big) \leq u.$$

Cardinality and weight constraints respectively amount to constraints on count and sum aggregate functions.

1.1.4 Strong equivalence

A logic program Π_1 is said to be *equivalent* to a logic program Π_2 in the sense of answer set semantics if Π_1 and Π_2 have the same answer sets. Moreover, Π_1 is *strongly equivalent* to Π_2 if $\Pi_1 \cup \Pi$ has the same answer sets as $\Pi_2 \cup \Pi$ (in other words, $\Pi_1 \cup \Pi$ is equivalent to $\Pi_2 \cup \Pi$), for every logic program Π (Lifschitz et al. [2001]). The study of strong equivalence is important because it allows us to simplify a part of a logic program without looking at the rest of it. Here are some examples.

SE.1 The one-rule program $\Pi_1 = \left\{ \begin{array}{rcl} p & \leftarrow & q\, ,\ not\, q \end{array} \right.$ is strongly equivalent to the empty set, so removing the rule $p \leftarrow q\, ,\ not\, q$ from a program does not affect the answer sets of the program.

SE.2 The program $\Pi_2 = \left\{ \begin{array}{lll} p & \leftarrow & q \\ q & \leftarrow & \end{array} \right.$ is strongly equivalent to $\Pi_3 = \left\{ \begin{array}{lll} p & \leftarrow & \\ q & \leftarrow & \end{array} \right.$.

SE.3 The one-rule programs $\Pi_4 = \{p \leftarrow \mathit{not}\, q\}$ and $\Pi_5 = \{p \leftarrow\}$ have the same answer set $\{p\}$, so they are equivalent. However, they are not strongly equivalent. To see this, we take another one-rule program $\Pi = \{q \leftarrow\}$. Indeed, while $\Pi_4 \cup \Pi$ has the answer set $\{q\}$, the answer set of $\Pi_5 \cup \Pi$ is $\{p, q\}$. Moreover, $\Pi'_4 = \left\{ \begin{array}{lll} p & \leftarrow & q \\ p & \leftarrow & \mathit{not}\, q \end{array} \right.$ and Π_5 are not strongly equivalent either. This fact can be seen by adding $q \leftarrow p$ to each of the programs, and shows that $q \mathit{\ or\ not}\, q \leftarrow$ is not strongly equivalent to $\top$ (or the empty set).

SE.4 The programs $\Pi_6 = \{p \mathit{\ or\ not}\, q \leftarrow\}$ and $\Pi_7 = \{p \leftarrow q\}$ have the same answer set $\emptyset$, but they are not strongly equivalent. Using item (III) and item (v) in the section of nested programs, we see that Π_6 is strongly equivalent to $\{p \leftarrow \mathit{not\ not}\, q\}$ and we know that double *not* does not cancel out at least when it is preceded literals. Alternatively, when we add to each program the rule $p \leftarrow \mathit{not}\, q$, we see that the resulting programs Π'_6 and Π'_7 are strongly equivalent respectively to $\Pi''_6 = \left\{ \ p \leftarrow \mathit{not\ not}\, q \mathit{\ or\ not}\, q \right.$ and $\Pi''_7 = \left\{ \ p \leftarrow q \mathit{\ or\ not}\, q \right.$ (see item (II) in the section of nested programs). However, Π''_6 and Π''_7 are not strongly equivalent because while the former is strongly equivalent to $\{p \leftarrow\}$, the later is not. Rest of the proof is included in the previous example.

SE.5 We have mentioned before the disjunctive logic program $\Pi_8 = \{p \mathit{\ or\ } q \leftarrow\}$ and the general logic program $\Pi_9 = \left\{ \begin{array}{lll} p & \leftarrow & \mathit{not}\, q \\ q & \leftarrow & \mathit{not}\, p \end{array} \right.$ have the same answer sets $\{p\}$ and $\{q\}$. However, they are not strongly equivalent. To see this, consider the program $\Pi_{10} = \left\{ \begin{array}{lll} p & \leftarrow & q \\ q & \leftarrow & p \end{array} \right.$. While $\{p, q\}$ is an answer set of $\Pi_8 \cup \Pi_{10}$, it is not an answer set of $\Pi_9 \cup \Pi_{10}$. On the other hand, when we consider another program $\Pi_{11} = \{\bot \leftarrow p\,,\, q\}$, we notice that $\Pi_8 \cup \Pi_{11}$ and $\Pi_9 \cup \Pi_{11}$ are strongly equivalent. This fact illustrates the possibility of eliminating 'exclusive disjunctions' from a logic program (Lifschitz et al. [2001]). We will apply the argument underlying the latter fact to programs containing strong negation in Subsection 1.2.3, and exemplify it with replacing q by $\sim p$.

1.2 Here-and-there (HT) logic

Here-and-there (**HT**) logic is obtained from *intuitionistic* logic that is characterised by the following axiom schemas

I1.	$\varphi \to (\psi \to \varphi)$	I2.	$(\varphi \wedge \psi) \to \varphi$
I3.	$(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$	I4.	$(\varphi \wedge \psi) \to \psi$
I5.	$(\varphi \to \psi) \to ((\varphi \to \chi) \to (\varphi \to (\psi \wedge \chi)))$	I6.	$\varphi \to (\varphi \vee \psi)$
I7.	$(\varphi \to \chi) \to ((\psi \to \chi) \to ((\varphi \vee \psi) \to \chi))$	I8.	$\psi \to (\varphi \vee \psi)$
I9.	$(\varphi \to \neg\psi) \to (\psi \to \neg\varphi)$	I10.	$\neg(\varphi \to \varphi) \to \psi$

and the inference rule of *modus ponens* through adding the axiom schema

$$(\neg\varphi \to \psi) \to (((\psi \to \varphi) \to \psi) \to \psi). \tag{1.15}$$

The latter characterises the 3-valued **HT** logic by Heyting (Heyting [1930]) and Gödel (Gödel [1932]). So, it is also known as Gödel's 3-valued logic, however it was apparently first axiomatised by Łukasiewicz (Łukasiewicz [1941]). The other well-known, but less used names for **HT** logic are the logic of *present-and-future* and the *Smetanich* logic.

Some other axioms may substitute Schema 1.15 though. One example is the Hosoi's axiom (Hosoi [1966])

$$\varphi \vee (\neg\psi \vee (\varphi \to \psi)). \tag{1.16}$$

However, we mainly consider one of the consequences of Schema 1.16, i.e.,

$$\neg\varphi \vee \neg\neg\varphi \tag{1.17}$$

in the proofs since it is more useful for our purposes. This schema is known as the *weak law of the excluded middle*, and can be derived from Schema 1.16 by simply taking ψ to be $\neg\varphi$. To sum up, **HT** logic is a non-classical monotonic logic, and it is the strongest intermediate logic between classical logic and intuitionistic logic. In other words, it strengths intuitionistic logic and is contained in classical logic.

HT models can be defined in terms of three truth values. These truth values were originally introduced by Heyting (Heyting [1930]) as a technical device for the purpose of demonstrating that intuitionistic logic is weaker than classical logic. Heyting remarks that the truth values in these tables can be interpreted as follows: 0 denotes a correct proposition, 1 denotes a false proposition, and 2 denotes a proposition that cannot be false, but whose correctness is not proved. On the other hand, intuitionistic logic cannot be described by a finite set of truth values (Gödel [1932]) and the proof of this fact uses an infinite monotonically decreasing sequence of systems whose first member is classical logic, and whose second member happens

to be **HT** logic. Moreover, **HT** logic properly contains all other intermediate logics in this infinite sequence.

Pearce was the first to realise that **HT** logic allows to characterise the answer set semantics of logic programs (Pearce [1996, 2006]) through a minimisation criterion over HT models. More recently, strong equivalence of logic programs was also characterised by means of **HT** logic (Lifschitz et al. [2001]). One of the applications of this result is that since **HT** is a 3-valued logic, deciding HT satisfiability is NP complete, and therefore not EXPTIME hard. Then, as a result of this, the strong equivalence of logic programs can also be verified in exponential time.

1.2.1 Language ($\mathcal{L}_{\mathbf{HT}}$)

The logical language to talk about **HT** logic as well as equilibrium logic, abbreviated by $\mathcal{L}_{\mathbf{HT}}$, is defined by the following grammar:

$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi,$$

where p ranges over a countably infinite set $\mathbb{P}$ of propositional variables. The last connective $\rightarrow$ has a strength between intuitionistic implication and material implication ($\supset$) of classical propositional logic. The other Boolean connectives are given in the usual manner: the negation $\neg\varphi$ is defined as $\varphi \rightarrow \bot$, in particular, $\neg\bot$ is given as $\bot \rightarrow \bot$ and $\top$ abbreviates both.

As usual, $\mathbb{P}_\varphi$ denotes the set of propositional variables occurring in an $\mathcal{L}_{\mathbf{HT}}$ formula φ, and this notation is also generalised to (finite) sets of such formulas, i.e., to HT theories.

1.2.2 HT models

We use the term *valuation* simply in the sense of a set of propositional variables. For instance, according to the valuation $V = \{p, q\}$, p and q are true and all other propositional variables are false.

An *HT model* is an ordered pair (H, T) of valuations H and T such that $H \subseteq T \subseteq \mathbb{P}$. Intuitively, such a pair describes 'two worlds': the first component *here* (H) is a set of true propositional variables and the second component *there* (T) is a set of non-false propositional variables which includes both true variables and variables whose truth cannot be proved. The latter are precisely contained in $T \setminus H$. Accordingly, in an HT model, 'here' is always included in 'there'. This inclusion is read as "H being *weaker* than T" (or alternatively, "T being *stronger* than H") and we call it the *heredity property* of HT models.

Given an HT model (H, T), the truth conditions are as follows:

$$\begin{array}{lll}
H,T \models_{\mathbf{HT}} p & \text{if} & p \in H, \text{ for every } p \in \mathbb{P}; \\
H,T \not\models_{\mathbf{HT}} \bot; & & \\
H,T \models_{\mathbf{HT}} \varphi \wedge \psi & \text{if} & H,T \models_{\mathbf{HT}} \varphi \text{ and } H,T \models_{\mathbf{HT}} \psi; \\
H,T \models_{\mathbf{HT}} \varphi \vee \psi & \text{if} & H,T \models_{\mathbf{HT}} \varphi \text{ or } H,T \models_{\mathbf{HT}} \psi; \\
H,T \models_{\mathbf{HT}} \varphi \to \psi & \text{if} & (H,T \not\models_{\mathbf{HT}} \varphi \text{ or } H,T \models_{\mathbf{HT}} \psi) \text{ and} \\
 & & (T,T \not\models_{\mathbf{HT}} \varphi \text{ or } T,T \models_{\mathbf{HT}} \psi).
\end{array}$$

Therefore, we infer that for every $p \in \mathbb{P}$:

$$\begin{array}{lll}
H,T \models_{\mathbf{HT}} \neg p & \text{if} & p \notin T; \\
H,T \models_{\mathbf{HT}} \neg\neg p & \text{if} & p \in T.
\end{array}$$

Equivalently, in **HT** logic the value assigned to a propositional variable p is determined by a three valued mapping $f : \mathbb{P} \to \{2\ (\textit{true}), 1\ (\textit{undefined}), 0\ (\textit{false})\}$ with incomplete information of the world:

$$f(p) = \begin{cases} 2 & \text{if} \quad p \in H \\ 1 & \text{if} \quad p \in T \setminus H \\ 0 & \text{if} \quad p \notin T \end{cases}$$

The function f is generalised to complex formulas in a natural way: while '$\wedge$' and '$\vee$' return respectively a maximum value and a minimum value, $f(\varphi \to \psi) = 2$ if $f(\varphi) \geq f(\psi)$ or returns $f(\psi)$ otherwise.

An HT model is said to be *total* if $H = T$. Therefore, a total HT model turns out to be classical. More explicitly, when H=T, the three valued interpretation is isomorphic to a two valued classical interpretation, i.e., $T,T \models_{\mathbf{HT}} \varphi$ in **HT** logic is the same as $T \models \varphi$ in classical logic. In particular, $T,T \models_{\mathbf{HT}} \varphi \to \psi$ in **HT** logic if and only if $T \models \varphi \supset \psi$ in classical logic, where $\supset$ refers to material implication.

When $H,T \models \varphi$ we say that (H,T) is an HT model of φ. A formula φ is *HT valid* if and only if every HT model is also an HT model of φ. For instance, while $\neg\varphi \vee \neg\neg\varphi$ is valid in **HT** logic (see Schema 1.17), $\varphi \vee \neg\varphi$ is not valid, neither is $\neg\neg\varphi \to \varphi$ nor is $((\varphi \to \psi) \to \varphi) \to \varphi$. A simple counterexample for the last two formulas is the HT model $(\emptyset, \{p\})$. Note that in fact $(\varphi \vee \neg\varphi) \leftrightarrow (\neg\neg\varphi \to \varphi)$ is HT valid (see Schema 1.20 below), which means that $\varphi \vee \neg\varphi$ and $\neg\neg\varphi \to \varphi$ are HT equivalent. Also, notice that adding any of these nonvalid schemas to **HT** logic would give us a two valued classical logic.

HT models have been studied since Gödel in order to give semantics to an implication with strength between intuitionistic and material implications. On one hand, the implication in $\mathcal{L}_{\mathbf{HT}}$ is interpreted in a non-classical way and is therefore different from material implication ($\supset$). Its truth condition is:

$$H,T \models_{\mathbf{HT}} \varphi \to \psi \quad \text{iff} \quad H,T \models_{\mathbf{HT}} \varphi \supset \psi \quad \text{and} \quad T,T \models_{\mathbf{HT}} \varphi \supset \psi,$$

where $\supset$ is interpreted just as in classical propositional logic. To spell it out, its truth condition is: "$H, T \models \varphi \supset \psi$" if and only if "$H, T \not\models \varphi$ or $H, T \models \psi$". Hence, material implication ($\supset$) here is just a shorthand enabling a concise formulation. On the other hand, **HT** logic also gives a slightly different semantics to its implication from that of intuitionistic logic. We have a good reason justifying this fact: it is easy to see that *De Morgan's laws*

$$\neg(\varphi \vee \psi) \leftrightarrow \neg\varphi \wedge \neg\psi \tag{1.18}$$

$$\neg(\varphi \wedge \psi) \leftrightarrow \neg\varphi \vee \neg\psi \tag{1.19}$$

are valid in **HT** logic, whereas although the first equivalence and one half of the second are intuitionistically provable, $\neg(\varphi \wedge \psi) \rightarrow \neg\varphi \wedge \neg\psi$ is not. Another interesting equivalence that is also valid in **HT** logic is

$$(\neg\varphi \vee \psi) \leftrightarrow (\neg\neg\varphi \rightarrow \psi). \tag{1.20}$$

However, while one half of this equivalence can be proved intuitionistically, right-to-left statement, i.e., $(\neg\neg\varphi \rightarrow \psi) \rightarrow (\neg\varphi \rightarrow \psi)$ is not a provable statement of intuitionistic logic. In each case, we consider two cases $\neg\varphi$ and $\neg\neg\varphi$ in order to prove, in **HT** logic, one half of the statements that are not proved intuitionistically. Now, we list some other equivalences of **HT** logic (Pearce [2006]):

$$\begin{array}{rcl}
\neg\neg\neg\varphi & \leftrightarrow & \neg\varphi \\
\varphi \rightarrow (\neg\neg\psi \vee \chi) & \leftrightarrow & (\neg\psi \wedge \varphi) \rightarrow \chi \\
(\neg\neg\varphi \wedge \psi) \rightarrow \chi & \leftrightarrow & \psi \rightarrow (\neg\varphi \vee \chi) \\
\Big((\varphi \vee \psi) \wedge \chi\Big) \rightarrow \alpha & \leftrightarrow & \left\{ \begin{array}{l} (\varphi \wedge \chi) \rightarrow \alpha \\ (\psi \wedge \chi) \rightarrow \alpha \end{array} \right\} \\
\varphi \rightarrow \Big((\psi \wedge \chi) \vee \alpha\Big) & \leftrightarrow & \left\{ \begin{array}{l} \varphi \rightarrow (\psi \vee \alpha) \\ \varphi \rightarrow (\chi \vee \alpha) \end{array} \right\} \\
\neg(\varphi \rightarrow \psi) & \leftrightarrow & \neg\neg\varphi \wedge \neg\psi \\
\Big((\varphi \rightarrow \psi) \wedge \chi\Big) \rightarrow \alpha & \leftrightarrow & \left\{ \begin{array}{rcl} (\neg\varphi \wedge \chi) & \rightarrow & \alpha \\ (\psi \wedge \chi) & \rightarrow & \alpha \\ \chi & \rightarrow & \varphi \vee \neg\psi \vee \alpha \end{array} \right\} \\
\varphi \rightarrow \Big((\psi \rightarrow \chi) \vee \alpha\Big) & \leftrightarrow & \left\{ \begin{array}{rcl} (\psi \wedge \varphi) & \rightarrow & (\chi \vee \alpha) \\ (\neg\chi \wedge \varphi) & \rightarrow & (\neg\psi \vee \alpha) \end{array} \right\}
\end{array}$$

We also give a bunch of results that will help us comment on the truth of complex formulas.

Lemma 1.2 *Given an HT model (H, T) and an $\mathcal{L}_{\mathbf{HT}}$ formula φ, we have:*

1. $H, T \models_{\mathbf{HT}} \varphi$ *implies* $T \models \varphi$;

2. $H, T \models_{\mathbf{HT}} \neg\varphi \quad \textit{iff} \quad T \not\models \varphi$;

3. $H, T \models_{\mathbf{HT}} \neg\neg\varphi \quad \textit{iff} \quad T \models \varphi$.

To begin with, one should note that the statements on the right hand side (RHS) of Lemma 1.2 refer to satisfaction in classical propositional logic. The first property (Lemma 1.2.1) is the *heredity property of intuitionistic logic*, adapted to HT logic, which says that if a formula has an HT model than it also has a total (or a classical) model. Lemma 1.2.1 is also known as the *monotonicity property* of HT logic. The intuition behind this property is obvious because it is guaranteed by the condition $H \subseteq T$ in any HT model (H, T), i.e., the heredity property of HT models.

An $\mathcal{L}_{\mathbf{HT}}$ formula φ is a *consequence* of a set Γ of formulas in **HT** logic (symbolically, $\Gamma \models_{\mathbf{HT}} \varphi$) if every HT model of Γ also satisfies φ. For instance, $p \vee \neg p$ is not a theorem of **HT** logic. To spell it out, it is not a consequence of the emptyset. Moreover, $\neg q \rightarrow p$ is not a consequence of $\neg p \rightarrow q$ either. In each case, $(\emptyset, \{p\})$ is a simple counter-example. Finally, we say that Φ and Ψ are *HT-equivalent* if they share exactly the same HT models.

We can claim as a consequence of the following lemma that the *finite model property* (perhaps better called a *finite valuation property*) holds for **HT** logic: if an $\mathcal{L}_{\mathbf{HT}}$ formula φ has an HT model then there also exists a pair of finite here and there sets (and hence an HT model (H, T)) such that $H, T \models \varphi$.

Lemma 1.3 *Given an $\mathcal{L}_{\mathbf{HT}}$ formula φ and a propositional variable $q \in \mathbb{P}$ such that $q \notin \mathbb{P}_\varphi$, we have*

$$H, T \models_{\mathbf{HT}} \varphi \quad \textit{iff} \quad H, T \cup \{q\} \models_{\mathbf{HT}} \varphi \quad \textit{iff} \quad H \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \varphi.$$

Proof. See Lemma A.1 and its proof in Section A.1 of Appendix A. q.e.d.

We can always generalise the proposition above into a handier version: for an $\mathcal{L}_{\mathbf{HT}}$ formula φ and $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\varphi = \emptyset$, we have $H, T \models \varphi$ if and only if $H \cup Q, T \cup P$ for every $Q \subseteq P$.

During the last decades, HT models have been further investigated by Pearce, Valverde, Cabalar, Lifschitz, Ferraris, and others as the basis of equilibrium logic which constitutes a semantical framework for **ASP** (Cabalar and Ferraris [2007]; Cabalar et al. [2007]; Ferraris et al. [2007]; Lifschitz [2010]; Lifschitz et al. [2001]; Pearce [1996]; Pearce et al. [2000]). We will deal with the subject in detail in the next section, but before we would like to discuss a striking feature of **HT** logic: it is able to capture the strong equivalence of logic programs.

1.2.3 Capturing strong equivalence in HT logic

We are interested in **HT** logic mostly because it provides a basis for the semantics of equilibrium logic which proposes an underlying logical framework for **ASP**.

However, another interesting feature of **HT** logic is to capture the strong equivalence concept of logic programming. As well as in **LP**, in nonmonotonic reasoning we talk about strong equivalence of theories Γ_1 and Γ_2 when $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ share the same models for any theory Γ. There is an exponential time algorithm for verifying strong equivalence.

Strong equivalence of programs without strong negation in HT logic
It has been proved in Lifschitz et al. [2001] that two programs Π_1 and Π_2, not including strong negation, are strongly equivalent if and only if they, viewed as sets of propositional formulas, are equivalent in **HT** logic. To see this, we need to translate a program containing rules that have the form of Definition 1.1 into HT theories containing corresponding implications in $\mathcal{L}_{\mathbf{HT}}$ (see Subsection 1.1.1). We denote an HT theory that corresponds to a program Π by Γ_Π. We first reexamine the examples given in Subsection 1.1.4.

SE′.1 $\Pi_1 = \{p \leftarrow q\,,\ not\, q\}$ and $\emptyset$ are strongly equivalent: indeed, the formula $q \wedge \neg q \rightarrow p$ is equivalent to $\top$ in **HT** logic.

SE′.2 $\Pi_2 = \left\{ \begin{array}{lll} p & \leftarrow & q \\ q & \leftarrow & \end{array} \right.$ and $\Pi_3 = \left\{ \begin{array}{lll} p & \leftarrow & \\ q & \leftarrow & \end{array} \right.$ are strongly equivalent because the HT theories $\{q \rightarrow p, q\}$ and $\{p, q\}$ are clearly equivalent.

SE′.3 $\Pi_4 = \{p\, or\, not\, q \leftarrow\}$ and $\Pi_5 = \{p \leftarrow q\}$ are not strongly equivalent: recall that the HT formula $p \vee \neg q$ is equivalent to $\neg\neg q \rightarrow p$ (see Schema 1.20 in Subsection 1.2.2), however it is easy to see that the latter is not HT equivalent to $q \rightarrow p$. To prove it formally, consider $(\emptyset, \{p, q\})$ which is an HT model of Π_5, but not of Π_4.

SE′.4 $\Pi_6 = \{p \leftarrow not\, q\}$ and $\Pi_7 = \{p \leftarrow\}$ are not strongly equivalent because $\neg q \rightarrow p$ and p are not HT equivalent. Note that $(\emptyset, \{q\})$ is a model for the former, but not for the latter. Moreover, $\Pi_6' = \left\{ \begin{array}{lll} p & \leftarrow & q \\ p & \leftarrow & not\, q \end{array} \right.$ and Π_7 are not strongly equivalent either. Notice that $(\emptyset, \{p, q\})$ is a model of $(q \rightarrow p) \wedge (\neg q \rightarrow p)$, but not of p. So, we see once again that **HT** logic provides us a very easy check for strong equivalence.

SE′.5 $\Pi_8 = \{p\, or\, q \leftarrow\}$ and $\Pi_9 = \left\{ \begin{array}{lll} p & \leftarrow & not\, q \\ q & \leftarrow & not\, p \end{array} \right.$ are not strongly equivalent because the HT theories $\{p \vee q\}$ and $\{\neg q \rightarrow p, \neg p \rightarrow q\}$ are not equivalent. Note that $(\emptyset, \{p, q\})$ is a model of the latter, but not for the former. Moreover, it is now easier to see that given $\Pi_{11} = \{\bot \leftarrow p\,,\ q\}$, $\Pi_8 \cup \Pi_{11}$ and $\Pi_9 \cup \Pi_{11}$ are strongly equivalent because the formulas $(p \vee q) \wedge (\neg p \vee \neg q)$ and $(\neg q \rightarrow$

$p) \wedge (\neg p \rightarrow q) \wedge (\neg p \vee \neg q)$ are HT equivalent. Recall that De Morgan's laws are valid in **HT** logic, so $\neg(p \wedge q)$ is equivalent to $\neg p \vee \neg q$ (see Schema 1.18). Hence, for any model (H, T) of both formulas, when $p \notin T$ then $q \in H$, and vice versa.

SE′.6 Finally we verify, using HT models, that $p \, or \, not \, p \leftarrow$ and $p \leftarrow p$ are not strongly equivalent. This fact has been partly discussed in Subsection 1.1.2.2 in the **LP** context. In HT logic, we simply support this fact saying that $p \vee \neg p$ and $p \leftarrow p$ are not HT equivalent: while $(\emptyset, \{p\})$ is not a model of the former which is so not HT valid, the latter is HT valid.

The following is an interesting example because we discuss the strong equivalence of two programs in which they differ just by a single rule, but these distinguishing rules are not strongly equivalent.

SE.7 In any program containing the rules $\left\{ \begin{array}{rcl} & \leftarrow & p \, , \, q \\ p \, or \, q & \leftarrow & \\ p & \leftarrow & r \end{array} \right.$ replacing the last rule by the constraint $\leftarrow not \, p \, , \, r$ does not affect the programs answer sets because the corresponding sets of formulas are equivalent in **HT** logic. However, note that $p \leftarrow r$ and $\leftarrow not \, p \, , \, r$ are not strongly equivalent when they stand alone. Note that although they have the same answer set $\emptyset$, appending the rule $r \leftarrow$ to both rules results in the answer set $\{p, r\}$ for the former, yet the latter would not have an answer set anymore.

The last two examples use the HT equivalences described in Subsection 1.2.2 (see the schemas given in Table 1.2.2).

SE.8 The generalised disjunctive program $\Pi = \{q \, or \, not \, p \leftarrow r\}$ is not strongly equivalent to $\Pi' = \{q \leftarrow r \, , \, p\}$: the corresponding theories $\{r \rightarrow (\neg p \vee q)\}$ and $\{(r \wedge p) \rightarrow q\}$ are not HT equivalent since $(\{r\}, \{p, q, r\})$ is an HT model of the latter theory, but not for the former. Moreover, we know that $r \rightarrow (\neg p \vee q)$ is HT equivalent to $(r \wedge \neg\neg p) \rightarrow q$.

The other distinguishing feature of HT logic in search of strong equivalence is that it allows us to find out when programs with embedded implications (i.e., programs with conditional rules containing implication in the body) can be replaced by ordinary nested programs we have seen before in this chapter.

SE.9 It is easy to see that a conditional rule $r \leftarrow ((p \rightarrow q) \, , \, s)$ is not strongly equivalent to the ordinary nested rule $r \leftarrow ((not \, p \, or \, q) \, , \, s)$: the corresponding HT implication $((p \rightarrow q) \wedge s) \rightarrow r$ is not HT equivalent to $((\neg p \vee q) \wedge s) \rightarrow r$. Note that $(\{s\}, \{p, q, r, s\})$ is an HT model of the latter, but not of the former. Alternatively, recall that while the latter HT formula is HT equivalent

to the theory $\Gamma = \{(\neg p \wedge s) \to r, (q \wedge s) \to r\}$, the former HT formula is HT equivalent to the theory $\Gamma \cup \{s \to (p \vee \neg q \vee r)\}$.

Similarly, we can also discuss strong equivalence when implication is allowed in the heads of rules. Note that the implication we mention here is different from the implication we have seen as an abbreviation in the language of ordinary nested programs.

Strong equivalence of arbitrary programs in HT logic
Questions concerning strong equivalence also arise for programs containing strong negation. In the extended setting, any two programs Π_1 and Π_2 are strongly equivalent if and only if $\widetilde{\Pi}_1 \cup Cons$ and $\widetilde{\Pi}_2 \cup Cons$, when viewed as HT theories, are equivalent in **HT** logic. In particular, when such programs do not include strong negation $\Pi_i = \widetilde{\Pi}_i$ for $i = 1, 2$ and $Cons$ reduces to $\emptyset$. For example, a one-rule program $\Pi = \{p \, or \sim p\}$ is strongly equivalent to

$$\Pi' = \left\{ \begin{array}{rcl} p & \leftarrow & not \sim p \\ \sim p & \leftarrow & not \, p \end{array} \right.$$

(Erdem and Lifschitz [1999]). To see this, we first need to construct

$$\widetilde{\Pi} = \left\{ \; p \, or \, \widetilde{p} \leftarrow \right. \quad \text{and} \quad \widetilde{\Pi}' = \left\{ \begin{array}{rcl} p & \leftarrow & not \, \widetilde{p} \\ \widetilde{p} & \leftarrow & not \, p \end{array} \right. \quad \text{and} \quad Cons = \left\{ \leftarrow p \, , \, \widetilde{p} \; . \right.$$

Then, we follow the same procedure for $\widetilde{\Pi} \cup Cons$ and $\widetilde{\Pi}' \cup Cons$ as we followed in the previous part. Hence, the rest is to show that $(p \vee \widetilde{p}) \wedge (\neg p \vee \neg \widetilde{p})$ and $(\neg \widetilde{p} \to p) \wedge (\neg p \to \widetilde{p}) \wedge (\neg p \vee \neg \widetilde{p})$ which is exactly the same work, with q substituted for $\widetilde{p}$, as the last example of SE′.5 in the previous section.

1.2.4 Least extension of HT logic: $\mathbf{N_5}$

In this subsection, we briefly discuss an extension of **HT** logic with strong negation ($\sim$). Adding $\sim$ to the language of **HT** logic, as well as the Vorob'ev axioms

N1.	$\sim(\varphi \to \psi) \leftrightarrow \varphi \wedge \sim\psi$	N2.	$\sim(\varphi \wedge \psi) \leftrightarrow \sim\varphi \vee \sim\psi$
N3.	$\sim(\varphi \vee \psi) \leftrightarrow \sim\varphi \wedge \sim\psi$	N4.	$\sim\sim\varphi \leftrightarrow \varphi$
N5.	$\sim\neg\varphi \leftrightarrow \varphi$	N6.	(for atomic φ) $\sim\varphi \to \neg\varphi$

(where $\varphi \leftrightarrow \psi$ abbreviates $(\varphi \to \psi) \wedge (\psi \to \varphi)$) gives a 5-valued monotonic formalism, called *here-and-there logic with strong negation*. In the literature, it is mainly known as Nelson's constructive logic **N** (see Appendix B.1.1). Following Pearce's notation, we denote it by $\mathbf{N_5}$ where "5" stresses the number of truth

values. Axioms N1-N5 allow the usual normal form transformation known from classical logic, by elimination of double negation and De Morgan's laws, in order to move negations in front of atomic formulas only. By N6, we see why $\sim$ is called strong negation: strong negation is indeed stronger than intuitionistic negation.

$\mathbf{N_5}$ is the least strong negation extension of **HT** logic and so it is a conservative extension of **HT** logic. So, given an N_5 formula φ without strong negation,

$$\varphi \text{ is a theorem of } \mathbf{HT} \text{ logic} \quad \text{if and only if} \quad \varphi \text{ is a theorem of } \mathbf{N_5}.$$

This extension also has many key metalogical properties of **HT** logic (see Kracht [1998] for a detailed study).

Like HT models, N_5 models are ordered pairs (H, T) such that $H \subseteq T$, except that H and T are now sets of literals ($H, T \subseteq \mathit{Lit}$) which are consistent. One exception is $H = T = \mathit{Lit}$. Given an N_5 model (H, T), the truth conditions are inductively extended using the following additional clauses:

$$\begin{array}{lll}
H,T \models_{\mathbf{N_5}} \sim(\varphi \wedge \psi) & \text{if} & H,T \models_{\mathbf{N_5}} \sim\varphi \text{ or } H,T \models_{\mathbf{N_5}} \sim\psi; \\
H,T \models_{\mathbf{N_5}} \sim(\varphi \vee \psi) & \text{if} & H,T \models_{\mathbf{N_5}} \sim\varphi \text{ and } H,T \models_{\mathbf{N_5}} \sim\psi; \\
H,T \models_{\mathbf{N_5}} \sim(\varphi \to \psi) & \text{if} & H,T \models_{\mathbf{N_5}} \varphi \text{ and } H,T \models_{\mathbf{N_5}} \sim\psi;
\end{array}$$

$$H,T \models_{\mathbf{N_5}} \sim\neg\varphi \quad \text{iff} \quad H,T \models_{\mathbf{N_5}} \sim\sim\varphi \quad \text{iff} \quad H,T \models_{\mathbf{N_5}} \varphi.$$

From these primitives we infer some interesting results: (note that the model and the validity concepts are defined as before)

1. A restricted form of contraposition holds for $\neg$ in the sense of logical equivalence, but not for $\sim$: while $(\varphi \to \psi) \to (\neg\psi \to \neg\varphi)$ is HT valid, it is not valid anymore when implication turned backwards. A simple counter example is the HT model $(\{p\}, \{p, q\})$ for the formulas $p \to q$ and $\neg q \to \neg p$. However, $(\varphi \to \psi) \to (\sim\psi \to \sim\varphi)$ is not N_5 valid. For instance, $\{\sim q\}$ is a model for $p \to q$, whereas it is not a model for $\sim q \to \sim p$.

2. The law of double negation holds for $\sim$, but not for $\neg$: while $\sim\sim\varphi \leftrightarrow \varphi$ is N_5 valid, $\neg\neg\varphi \leftrightarrow \varphi$ is not HT valid, accordingly not N_5 valid either.

We also mention shortly here an extension of $\mathbf{N_5}$: adding the axiom schema

$$\neg\neg\varphi \to \varphi$$

to $\mathbf{N_5}$ yields a 3-valued extension of this logic, which is denoted by $\mathbf{N_3}$. It is also called *classical logic with strong negation* since it is a conservative extension of classical logic. This logic is important since it is precisely the logic where total N_5 models appear. Note that the concept of an equilibrium model (see Section 1.3) in $\mathbf{N_5}$ is defined analogously to the case of **HT** logic (see Pearce [2006] for a detailed study), so $\mathbf{N_3}$ is used in the equilibrium model construction.

Lastly, strong equivalence of arbitrary logic programs can be alternatively studied in $\mathbf{N_5}$: two arbitrary programs Π_1 and Π_2 are strongly equivalent if and only if they, when viewed as the theories of $\mathbf{N_5}$, are equivalent in $\mathbf{N_5}$.

1.3 Equilibrium logic

Equilibrium logic is a general purpose nonmonotonic reasoning closely associated with **ASP**. In particular, equilibrium logic generalises all previous syntax extensions of **ASP** except epistemic specifications by Gelfond (Gelfond [2011, April 28]) to full propositional logic and admits a natural extension to the first order case. Moreover, it provides a new and natural logical characterisation of answer sets as a form of minimal model reasoning built upon **HT** logic, and on its least extension $\mathbf{N_5}$ which are maximal logics capturing strong equivalence. However, throughout this work, we will mainly follow **HT** logic as the base logic.

The alternative characterisation of answer sets through equilibrium models is much easier to work with because it has the advantage of being fully 'declarative', or 'logical', and does not involve any procedural (or operational) fixed point construction, and instead uses a simple definition of minimal model reasoning, more in the style of circumscription. Moreover, equilibrium logic extends the concept of an answer set from logic programs to arbitrary sets of formulas giving a logical interpretation by means of a translation of rules like "$p \, or \sim q \leftarrow r \, , \, not \, s$" into a standard logical notation "$r \wedge \neg s \rightarrow p \vee \sim q$" where $\sim$ refers to strong negation. As defined before, logic programs correspond to a special case of theories in which every formula is an implication: the antecedent is the body and the consequent is the head of the corresponding rule. The other advantage of equilibrium logic beyond **ASP** is that it allows some logical techniques, for instance, methods from many valued semantics such as tableaux, signed logics, etc. Briefly, equilibrium logic captures and extends **ASP** in a full success.

1.3.1 Equilibrium logic based on HT logic

In this subsection, we introduce equilibrium logic without strong negation, i.e., when it is based on **HT** logic and call it simply equilibrium logic.

Expectedly, equilibrium logic shares the same language with **HT** logic, and its semantics is given through equilibrium models that are defined in an indirect way based on **HT** logic. Roughly speaking, an equilibrium model of a formula is a classical model satisfying a minimality condition with respect to HT models. However, the selection of such minimal models gives a nonmonotonic character to equilibrium logic. Formally, an equilibrium model of φ is a set of propositional variables $T \subseteq \mathbb{P}$ such that

1. $T \models \varphi$ in classical propositional logic, and

2. there is no $H \subset T$ such that $H, T \models_{\mathbf{HT}} \varphi$.

Observe that the first condition '$T \models \varphi$ in classical propositional logic' can be replaced by '$T, T \models_{\mathbf{HT}} \varphi$ in **HT** logic', and the second condition is known as the *minimality condition* with respect to HT models. It is easy to see that all HT equivalent formulas have the same equilibrium models.

Here are some examples.

- First, $T = \emptyset$ is an equilibrium model of $\neg p$ because
  - (1) for the HT model $(\emptyset, \emptyset)$ we have $\emptyset, \emptyset \models_{\mathbf{HT}} \neg p$, and
  - (2) there is no set H that is strictly included in the empty set (i.e., when $T = \emptyset$ the second condition is trivially satisfied).

  Moreover, $T = \emptyset$ is the only equilibrium model of $\neg p$. To see this, suppose T is an equilibrium model of $\neg p$ for some $T \neq \emptyset$. T cannot contain p, otherwise condition (1) would be violated. Therefore T contains q for some $q \neq p$, but then condition (2) is violated since $\emptyset, T \models_{\mathbf{HT}} \neg p$.

- The empty set is also the unique equilibrium model of all HT valid formulas like $\top$. Moreover, formulas such as $p \to q$, $p \to \neg q$ and $\neg p \to \neg q$ have also a unique equilibrium model, i.e., $\emptyset$.

- Then, for $q \neq p$, $\{q\}$ is a unique equilibrium model of $\neg p \to q$ because $\{q\}, \{q\} \models_{\mathbf{HT}} \neg p \to q$ and $\emptyset, \{q\} \not\models_{\mathbf{HT}} \neg p \to q$. However, since both $\{p\}, \{p\} \models_{\mathbf{HT}} \neg p \to q$ and $\emptyset, \{p\} \models_{\mathbf{HT}} \neg p \to q$ hold, the set $\{p\}$ is not an equilibrium model of this formula, and neither is the empty set because $\emptyset, \emptyset \not\models_{\mathbf{HT}} \neg p \to q$. When $q = p$ the formula has no equilibrium models.

It is interesting to observe that while $\neg p \to p$ has no equilibrium model, $T = \emptyset$ is the unique equilibrium model of $p \to \neg p$. This is probably a special case of a more general observation on equilibrium models: although $p \to q$ and $\neg p \vee q$ are not HT-equivalent, they are equivalent in the sense of equilibrium models.

- Next, while the unique equilibrium models of $p \vee q$ are $\{p\}$ and $\{q\}$, $p \wedge q$ has exactly one equilibrium model $\{p, q\}$.

- The HT formula $\neg\neg p \to p$ has (only) two equilibrium models: $\emptyset$ and $\{p\}$. Using Schema 1.20, we conclude that $p \vee \neg p$ and $\neg\neg p \to p$ are HT equivalent. Therefore, the former formula has also the same equilibrium models.

- Finally, $\bot$ and $\neg\neg p$, for every $p \in \mathbb{P}$, have no equilibrium models. In fact, $\bot$ has no HT model either. As for $\neg\neg p$, it is HT equivalent to $\neg\neg p \vee p$ (or $\neg p \to p$) and we mention above that the last has no equilibrium models. Alternatively, to prove this fact, we can also say that even if the first condition is satisfied as in $T = \{p\}$, the minimality condition fails since $\emptyset, \{p\} \models_{\mathbf{HT}} \neg\neg p$.

Remark 1.1 *As an immediate outcome of Lemma 1.3, we now remark on equilibrium models of an $\mathcal{L}_{\mathbf{HT}}$ formula φ. One should note that an equilibrium model T of φ should be a subset of $\mathbb{P}_\varphi$, if exists; otherwise $T, T \models \varphi$ and $T \setminus \{q\}, T \models \varphi$ for some $q \in T \setminus \mathbb{P}_\varphi$ both hold by Lemma 1.3 and that would contradict the minimality condition of equilibrium models. Consequently, an equilibrium model of a formula is always finite, thus so is the number of its equilibrium models. Finally, $\neg\varphi \in \mathcal{L}_{\mathbf{HT}}$ has either a unique equilibrium model $\emptyset$ (see Proposition) or none. As a result of this fact, if $T = \emptyset$ is an equilibrium model for φ then $\neg\varphi$ has no equilibrium model.*

We now define the notion of *equilibrium entailment*, and denote it by $\mid\!\approx$. However, the widely accepted symbol for this notion is $\mid\!\sim$: given $\mathcal{L}_{\mathbf{HT}}$ formulas φ and χ, we say that φ is a *consequence* of χ in equilibrium logic if for every equilibrium model T of χ, (T, T) is an HT model of φ. However, there are exceptions. When φ has no equilibrium models, we write $\varphi \mid\!\approx \psi$ if $\varphi \models_{\mathbf{HT}} \psi$. Moreover, $\mid\!\approx \psi$ if $\models_{\mathbf{HT}} \psi$. Here are some examples.

- Relations here hold both way, so related formulas are equivalent in the sense of equilibrium models: $\top \mid\!\approx \neg p$, $q \mid\!\approx \neg p \to q$, $\neg q \mid\!\approx p \to q$ and for $p \neq q$, $p \vee q \mid\!\approx (p \vee q) \wedge \neg(p \wedge q)$.
- $\bot \mid\!\approx \varphi$ does not necessarily hold in the reverse order. In fact, the other way around just holds when φ is HT equivalent to $\bot$.
- $\neg\neg p \mid\!\approx \varphi$, for every $\varphi \in \mathcal{L}_{\mathbf{HT}}$ such that $\|\varphi\|_{\mathbf{HT}} = \{(H, T) \ : \ p \in T\}$.
- These relation can be given just in one way, the converses are certainly false: $p \wedge q \mid\!\approx p \vee q$, and for $p \neq q$, $p \mid\!\approx \neg q$, but $\neg q \not\mid\!\approx p$ and $p \vee q \not\mid\!\approx p \wedge q$.
- $\mid\!\approx \varphi$ where φ is HT valid as in $\mid\!\approx \neg\neg p \to p$.

We then generalise this definition to HT theories ($\Gamma \mid\!\approx \varphi$) in the same way.

1.3.2 Equilibrium logic based on $\mathbf{N_5}$ logic

In this subsection, we present a slight extension of equilibrium logic we have introduced in Subsection 1.3.1. This extension includes strong negation and so is based

on $\mathbf{N_5}$. We continue calling this extension briefly equilibrium logic. The definition of equilibrium model can be easily generalised to this version, so we just give some examples to clarify it. However, one can refer to Pearce [2006] for a detailed study. Here are some examples.

- The N_5 theory $\Gamma = \{\sim q, \neg p \to q\}$ has no equilibrium models: one should first note that the candidate equilibrium models should certainly include $\sim q$, but $(\{\sim q\}, \{\sim q\}) \not\models_{\mathbf{N_5}} \neg p \to q$ since q is not included in the here world $\{\sim q\}$. We cannot add q into $\{\sim q\}$ because otherwise it would be inconsistent. So, the last option is to put p into $\{\sim q\}$. However, the second candidate model $\{p, \sim q\}$ has a smaller model $(\{\sim q\}, \{p, \sim q\})$ satisfying Γ and this violates the minimality condition.

- Recall that the HT formulas $\neg p \to q$ and $\neg q \to p$ both have a unique equilibrium model, respectively $\{q\}$ and $\{p\}$, but the N_5 formulas $\sim p \to q$ and $\sim q \to p$ have the same unique equilibrium model, i.e., $\emptyset$.

- Recall that the HT formula $p \vee \neg p$ has two equilibrium models: $\emptyset$ and $\{p\}$. However, the N_5 formula $p \vee \sim p$ behaves like $p \vee q$ and has two equilibrium models, namely $\{p\}$ and $\{\sim p\}$.

The equilibrium consequence relation is defined as before. In this subsection, we list some general properties of $\mid\!\approx$ as a nonmonotonic inference relation (see Pearce [2006] for the proofs).

- *reflexivity*: if $\varphi \in \Gamma$ then $\Gamma \mid\!\approx \varphi$.

- *cut*: if for every $i \in I$, $\Gamma \mid\!\approx \psi_i$ and $\Gamma \cup \{\psi\}_i \mid\!\approx \varphi$ then $\Gamma \mid\!\approx \varphi$.

- if $\Gamma \models_{\mathbf{N_5}} \varphi$ or $\Gamma \models_{\mathbf{N_5}} \psi$ then $\Gamma \mid\!\approx \varphi$ and $\Gamma \mid\!\approx \psi$ implies $\Gamma \cup \{\varphi\} \mid\!\approx \psi$.

- if $\Gamma \mid\!\approx \neg\varphi$ and $\Gamma \mid\!\approx \psi$ then $\Gamma \cup \{\neg\varphi\} \mid\!\approx \psi$.

- *disjunction in the antecedent*: if $\Gamma \cup \{\varphi\} \mid\!\approx \chi$ and $\Gamma \cup \{\psi\} \mid\!\approx \chi$ then $\Gamma \cup \{\varphi \vee \psi\} \mid\!\approx \chi$.

- *conditionalisation*: if $\Gamma \cup \{\varphi\} \mid\!\approx \psi$ then $\Gamma \mid\!\approx \varphi \to \psi$.

- *weak rationality*: if $\Gamma \mid\!\approx \psi$ and $\Gamma \cup \{\varphi\} \mid\!\approx \neg\psi$ then $\Gamma \mid\!\approx \neg\varphi$.

- *inverted form of weak rationality*: if $\Gamma \cup \{\varphi\} \mid\!\approx \psi$ and $\Gamma \mid\!\approx \neg\psi$ then $\Gamma \mid\!\approx \neg\varphi$

- *modus tollens* (for $\neg$): if $\Gamma \mid\!\approx \varphi \to \psi$ and $\Gamma \mid\!\approx \neg\psi$ then $\Gamma \mid\!\approx \neg\varphi$.

However, the following properties fail as a result of the nonclassical nature of the underlying logic $\mathbf{N_5}$ (or **HT**).

- *cautious monotonity*, i.e., the property of 'if $\Gamma \mid\!\approx \varphi$ and $\Gamma \mid\!\approx \psi$ then $\Gamma \cup \{\varphi\} \mid\!\approx \psi$' fails. So, $\mid\!\approx$ is not a cumulative inference relation.
- *proof by cases*, i.e., the property of 'if $\Gamma \cup \{\varphi\} \mid\!\approx \psi$ and $\Gamma \cup \{\sim\varphi\} \mid\!\approx \psi$ then $\Gamma \mid\!\approx \psi$' fails. In fact it is not a valid principle of constructive reasoning and fails also for $\mathbf{N_5}$. For a simple counter example, take $\Gamma = \emptyset$ and $\psi = \varphi \vee \sim\varphi$.
- *rationality* (or *rational monotony*), i.e., the property of 'if $\Gamma \mid\!\approx \psi$ and $\Gamma \cup \{\varphi\} \not\mid\!\approx \psi$ then $\Gamma \mid\!\approx \neg\varphi$', fails: give an example.
- *strong rationality*, i.e., the property of 'if $\Gamma \mid\!\approx \psi$ and $\Gamma \cup \{\varphi\} \not\mid\!\approx \psi$ then $\Gamma \mid\!\approx \sim\varphi$', fails.

Since $\mathbf{N_5}$ is a conservative extension of **HT** logic, expectedly the above-mentioned results, in which strong negation does not appear, also hold for equilibrium logic based on **HT** logic.

1.3.3 Relation to answer sets

This subsection gives a relation between **ASP** and equilibrium logic based on **HT** logic, but note that a similar relation can also be studied for equilibrium logic based on $\mathbf{N_5}$. One can refer to Pearce [2006] to see the latter. Therefore, in this work we will follow the previous approach and eliminate strong negation in logic programs as we have done in Subsection 1.2.3 (see also Lemma 1.1 in 1.1.2.2) in order to embed the answer set concept into equilibrium logic.

Answer sets are a special case of equilibrium models. Rules in **ASP** are similar to propositional formulas of **HT** logic, $\leftarrow$ and *not* being the counterparts of implication and negation. Introducing $\sim$ in **ASP** is similar to adding strong negation to **HT** logic. The following proposition illustrates this fact.

Proposition 1.1 *For any program* Π *and consistent set* $S \subset Lit$*, the following are equivalent:*

i. S *is an answer set of* Π*.*

ii. $\widetilde{S}$ *is a stable model of* $\widetilde{\Pi} \cup Cons$*.*

iii. $\widetilde{S}$ *is an equilibrium model of* $\Gamma_{\widetilde{\Pi} \cup Cons}$*.*

Note that (i)-(ii) is given for extended programs in Subsection 1.1.2.2 (see Lemma 1.1). The following examples uses Proposition 1.1 above: given

$$\Pi = \left\{ \begin{array}{rcl} p \, or \sim p & \leftarrow & \\ q & \leftarrow & p \\ \sim q & \leftarrow & \end{array} \right.$$

we have:

$$\widetilde{\Pi} = \left\{ \begin{array}{rcll} p \, or \, \widetilde{p} & \leftarrow & & \\ q & \leftarrow & p & \\ \widetilde{q} & \leftarrow & & \end{array} \right. \quad \text{and} \quad Cons = \left\{ \begin{array}{rcl} \leftarrow & p \, , \, \widetilde{p} \\ \leftarrow & q \, , \, \widetilde{q} \end{array} \right. .$$

The unique answer set of Π is $\{\sim p, \sim q\}$, and accordingly the only stable model of $\widetilde{\Pi} \cup Cons$ is $\{\widetilde{p}, \widetilde{q}\}$. Hence, it turns out that the latter is also the unique equilibrium model of

$$\Gamma_{\widetilde{\Pi} \cup Cons} = \{p \vee \widetilde{p}, p \rightarrow q, \widetilde{q}, \neg(p \wedge \widetilde{p}), \neg(q \wedge \widetilde{q})\}.$$

Recall that the one-rule program $p \, or \, not \, p \leftarrow$ has two nonminimal models: $\emptyset$ and $\{p\}$. On the other side of the discussion, the HT formula $p \vee \neg p$ has the same equilibrium models. Notice that equilibrium models do not follow in general a minimality condition according to set inclusion as it is in this example because otherwise they would not capture the answer sets of a program with *not* occurring in the head, i.e., generalised disjunctive programs, nor the answer sets of nested programs. However, the minimality condition in the definition of equilibrium models refer to the minimality according to a special partial ordering as follows: for every HT models (H, T) and (H', T'),

$$(H, T) \trianglelefteq (H', T') \;\text{ if }\; T = T' \text{ and } H \subseteq H'. \tag{1.21}$$

In this respect, the equilibrium models of $p \vee \neg p$, when viewed as total models $(\emptyset, \emptyset)$ and $(\{p\}, \{p\})$, are minimal according to $\trianglelefteq$ since $(\emptyset, \emptyset) \not\trianglelefteq (\{p\}, \{p\})$. At this point we find it useful to give an alternative equilibrium model definition (Pearce [2006]): given $T \subseteq \mathbb{P}$ and $\varphi \in \mathcal{L}_{\mathbf{HT}}$, T is an equilibrium model of φ if (T, T) is minimal under $\trianglelefteq$ among all HT models of φ. It is easy to see that both definitions overlap. This definition is then generalised to HT theories straightforwardly.

Before we pass to the next section, we summarise the main complexity results related with **HT** logic and equilibrium logic: each row associates a complexity class for a decision problem w.r.t. propositional HT theories.

Overall Complexity Results	
	HT Theories
HT model existence	NP
Equilibrium model existence	Σ_2^P
Equilibrium consequence	Π_2^P
HT equivalence	Π_2^P
Uniform equivalence	Π_2^P
Strong equivalence	$coNP$

1.4 Modal extension of logic programs: epistemic specifications (E-S)

In the beginning of the 90s, Gelfond has extended disjunctive logic programming by epistemic notions such as knowledge and belief (Gelfond [1991, 1994, 2011]). His *epistemic specifications* (**E-S**) allow to correctly reason about incomplete information in situations when there are multiple answer sets, alias belief sets because the notion of answer set was not powerful enough to deal with commonsense reasoning. The idea is to include a notion of introspection by means of two epistemic operators K and M ranging over all possible answer sets of a program. So, their meanings depend on not only the current answer set, but also on the collection of all answer sets. In other words, the presentation of epistemic operators into the language has made it possible to handle all belief sets determined by a program altogether rather than separately. The semantics of an epistemic specification generalises the answer set semantics for disjunctive logic programs: it is collections of all possible answer sets called *world views*. This operational semantics is defined by means of a transformation that eliminates K and M operators in the body of rules, resulting in a disjunctive logic program.

Since epistemic specifications were introduced, they have been refined once by Gelfond (Gelfond [2011]), and more recently Kahl has proposed a further improvement (Kahl [2014]). However, it seems that a fully satisfactory semantics has not been given yet. Some authors have undertaken to generalise the semantics of **E-S** (Chen [1997]; Truszczyński [2011]; Wang and Zhang [2005]). Moreover, none of them considers Gelfond's corrected version Gelfond [2011] of the formalism but rather the first, somewhat outdated version of Gelfond [1991, 1994]. In this section we recall the latest version of the language and the semantics of **E-S** proposed by Gelfond in Gelfond [2011], and we mainly consider propositional case. We also point out the differences with Kahl's approach.

1.4.1 Language ($\mathcal{L}_{\text{E-S}}$)

The language of epistemic specifications ($\mathcal{L}_{\text{E-S}}$) is an extension of the language of disjunctive logic programs (see Subsection 1.1.2.2) by the epistemic modal operators K and M (yet, just K is primitive) to respectively characterise knowledge and belief. The intuitive meanings of these operators are as follows: $\mathsf{K}\,\varphi$ and $\mathsf{M}\,\varphi$ respectively stand for "φ is known to be true" and "φ may be believed to be true"[1]. The language $\mathcal{L}_{\text{E-S}}$ also extends the literal concept. While we have used it so far in the sense of a propositional variable and its strong negation, from now on literals of this extended language are composed of two kinds (Gelfond [2011, April 28]):

- *objective literals*, abbreviated by l, are (non-modal) atomic formulas of the form $p(\bar{t})$[2] and $\sim p(\bar{t})$ where $\sim$ refers to strong negation (see Appendix B.1.1). $\mathbb{O}$-*Lit* denotes the set of all objective literals in $\mathcal{L}_{\text{E-S}}$ and it exactly corresponds to the set *Lit* of the language of disjunctive logic programs.

- *subjective literals*, abbreviated by g, are (modal) atomic formulas that have the form of $\mathsf{K}\,l$, $\mathsf{M}\,l$, $\sim\mathsf{K}\,l$ and $\sim\mathsf{M}\,l$ where l is an objective literal possibly preceded by negation as failure (*not*). We denote the set of all subjective literals of $\mathcal{L}_{\text{E-S}}$ by $\mathbb{S}$-*Lit*.

Formally, the literals of $\mathcal{L}_{\text{E-S}}$ are defined by the following grammar:

$$\begin{aligned} l &::= p \mid \sim p \\ g &::= \mathsf{K}\,l \mid \sim\mathsf{K}\,l \mid \mathsf{K}\,not\ l \mid \sim\mathsf{K}\,not\ l \mid \\ &\quad (\mathsf{M}\,l \mid \sim\mathsf{M}\,l \mid \mathsf{M}\,not\ l \mid \sim\mathsf{M}\,not\ l) \end{aligned}$$

where p ranges over a (fixed) countably infinite set of propositional variables $\mathbb{P}$. We have two kinds of negations: strong negation $\sim$ and default negation *not*. The modal operators K and M are *dual* of each other: M can be expressed in terms of K as follows[3]: for $l \in \mathbb{O}$-*Lit*,

$$\begin{aligned} \mathsf{M}\,l &\stackrel{\mathsf{def}}{=} \sim\mathsf{K}\ not\ l \quad (\dagger), \\ \mathsf{M}\ not\ l &\stackrel{\mathsf{def}}{=} \sim\mathsf{K}\,l \quad (\dagger\dagger). \end{aligned}$$

One should note that this restricted transformation is sufficient for our purposes, and allows us to use both operators interchangeably. Therefore, we choose K as a primitive in the above-mentioned grammar defining literals.

[1] And not "φ is compatible with the agent's belief", which would be the more standard reading of a modal operator that—as we will see below—is dual to the knowledge operator K.

[2] Objective literals with variables are just as shorthands for the collections of their ground instances.

[3] Note that $\mathsf{M}\,\varphi$ is not equivalent to $\sim\mathsf{K}\sim\varphi$.

An **E-S** *rule* ρ is of the form

$$\rho = l_1 \ or \ \ldots \ or \ l_k \leftarrow (g_{k+1} \ , \ \ldots \ , \ g_m) \ , \ (not \ l_{m+1} \ , \ \ldots \ , \ not \ l_n)$$

where $l_i \in \mathbb{O}\text{-}Lit$ for $i{=}1, \ldots, k$ and for $i{=}m+1, \ldots, n$ and $g_j \in \mathbb{O}\text{-}Lit \cup \mathbb{S}\text{-}Lit$ for $j{=}k+1, \ldots, m$. We note that Kahl's **E-S** rules differ only in subjective literals, which are in the form of $\mathsf{K}\, l$ and $\mathsf{M}\, l$, possibly preceded by default negation *not*. However, $\sim\mathsf{K}\, l$ and $not\, \mathsf{K}\, l$ have the same truth conditions in both approaches, so do $\sim\mathsf{M}\, l$ and $not\, \mathsf{M}\, l$. Finally, an *epistemic specification* T is a finite collection of **E-S** rules.

We call an $\mathcal{L}_{\textbf{E-S}}$ formula without modal operators *objective* (or *non-modal*), and an $\mathcal{L}_{\textbf{E-S}}$ formula in the form of $\mathsf{K}\, \varphi$, $\sim\mathsf{K}\, \varphi$, $\mathsf{M}\, \varphi$ or $\sim\mathsf{M}\, \varphi$ *subjective*.

Given a (epistemic) program $\Pi = \{R_k\}_{k \leq n}$, $\mathbb{P}_\Pi = \bigcup\limits_{k \leq n} \mathbb{P}_{R_k}$ denotes the set of variables that occur in this program which is simply a union of sets ($\mathbb{P}_{R_k}$) of all variables that exist in each rule R_k. On the other hand, $\mathbb{L}_\Pi = \bigcup\limits_{k \leq n} \mathbb{L}_{R_k}$ denotes the set of all (objective) literals that appear in this program. For example, for the epistemic specification Π:

$$\begin{array}{l} p \leftarrow \\ q \ or \sim r \leftarrow \\ s \leftarrow \mathsf{K} \sim p \\ \sim t \leftarrow \mathsf{M}\, q \end{array}$$

we have $\mathbb{P}_\Pi = \{p, q, r, s, t\}$ and $\mathbb{L}_\Pi = \{p, q, \sim r, s, \sim p, \sim t\}$.

1.4.2 World view semantics

A *simple theory* is either a consistent subset of $\mathbb{O}\text{-}Lit$ or $\mathbb{O}\text{-}Lit$ itself, and an E-S model is basically an ordered pair of the form $(\mathcal{S}, W)$ where $W \subseteq \mathbb{O}\text{-}Lit$ is a simple theory and $\mathcal{S} \subseteq 2^{\mathbb{O}\text{-}Lit}$ is a collection of such theories. However, W is not necessarily included in $\mathcal{S}$. Intuitively, $\mathcal{S}$ can be thought of a collection of possible belief sets of a reasoner while W represents his current (working) belief set.

The notions of truth ($\models_{\textbf{E-S}}$) and falsity ($\not\models_{\textbf{E-S}}$) are defined inductively over E-S models. The boolean cases are as usual, so the only truth conditions worth to

displaying are:[1] for $l \in \mathbb{O}\text{-}Lit$, $g \in \mathbb{S}\text{-}Lit$ and $\varphi \in \mathcal{L}_{\text{E-S}}$,

$$
\begin{array}{lll}
(\mathcal{S}, W) \models_{\text{E-S}} l & \text{if} & l \in W; \\
(\mathcal{S}, W) \models_{\text{E-S}} \mathit{not}\ l & \text{if} & l \notin W; \\
(\mathcal{S}, W) \models_{\text{E-S}} \mathsf{K}\, l & \text{if} & l \in S \text{ for every } S \in \mathcal{S}; \\
(\mathcal{S}, W) \models_{\text{E-S}} \mathsf{K}\ \mathit{not}\ l & \text{if} & l \notin S \text{ for any } S \in \mathcal{S}; \\
(\mathcal{S}, W) \models_{\text{E-S}} {\sim} g & \text{if} & \mathcal{S}, W \not\models_{\text{E-S}} g; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ l & \text{if} & {\sim} l \in W; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ \mathit{not}\ l & \text{if} & l \in W; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ \mathsf{K}\, l & \text{if} & l \notin S \text{ for some } S \in \mathcal{S}; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ \mathsf{K}\ \mathit{not}\ l & \text{if} & l \in S \text{ for some } S \in \mathcal{S}; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ {\sim} g & \text{if} & \mathcal{S}, W \models_{\text{E-S}} g; \\
(\mathcal{S}, W) \models_{\text{E-S}} {\sim} \varphi & \text{if} & (\mathcal{S}, W)_{\text{E-S}} =\!| \ \varphi.
\end{array}
$$

From such primitives, we infer that: for $l \in \mathbb{O}\text{-}Lit$,

$$
\begin{array}{lll}
(\mathcal{S}, W) \models_{\text{E-S}} \mathsf{M}\, l & \text{if} & l \in S \text{ for some } S \in \mathcal{S} \ \ (\text{ see } \dagger); \\
(\mathcal{S}, W) \models_{\text{E-S}} \mathsf{M}\ \mathit{not}\ l & \text{if} & l \notin S \text{ for some } S \in \mathcal{S} \ \ (\text{ see } \dagger\dagger); \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ \mathsf{M}\, l & \text{if} & l \notin S \text{ for any } S \in \mathcal{S}; \\
(\mathcal{S}, W)_{\text{E-S}} =\!| \ \mathsf{M}\ \mathit{not}\ l & \text{if} & l \in S \text{ for every } S \in \mathcal{S}.
\end{array}
$$

One crucial point to observe about the falsity notion is: $(\mathcal{S}, W)_{\text{E-S}} =\!| \ \varphi$ is not always equivalent to $(\mathcal{S}, W) \not\models_{\text{E-S}} \varphi$. One exception is when φ is an objective literal because we allow not only propositional variables, but also their strong negations into the valuations, and hence we present negations explicitly in the valuations. As a result, when an objective literal is the case, the former expression, i.e., falsifying an objective literal l amounts to including its negation ${\sim} l$ in the related valuation.

Another important point is: given an E-S model $(\mathcal{S}, W)$, negation as failure '*not*' works *locally* (on the second component) even if the second component changes in situations like "K *not* l" while the epistemic operators 'K' and 'M' always work globally (over the collection $\mathcal{S}$). Therefore, the truth (and the falsity) of an objective formula φ depends just on W, and so we simply write $W \models_{\text{E-S}} \varphi$. On the other hand, the truth (and the falsity) of a subjective formula ψ depends only on $\mathcal{S}$, and in this case we write, for short, $\mathcal{S} \models_{\text{E-S}} \psi$. We have mentioned before that $\mathsf{M}\, l$ is not equivalent to ${\sim}\mathsf{K}{\sim} q$. Indeed, for example, as $\mathcal{S} = \big\{\{p\}, \{p, {\sim} q\}\big\}$ satisfies ${\sim}\mathsf{K}{\sim} q$, it does not satisfy $\mathsf{M}\, q$.

From the discussion given above, we now obtain some equivalences: for every

[1] Since the double (strong) negation cancels out, expressions of the form ${\sim} l$ (where l is an objective literal) should not seem ambiguous to the reader.

collection $\mathcal{S}$ and $l \in \mathbb{O}$-*Lit*:

$$
\begin{array}{lll}
\mathcal{S} \models_{\textbf{E-S}} \mathsf{M}\, l & \text{iff} & \mathcal{S} \models_{\textbf{E-S}} {\sim}\mathsf{K}\, \mathit{not}\, l \\
\mathcal{S} \models_{\textbf{E-S}} \mathsf{M}\, \mathit{not}\, l & \text{iff} & \mathcal{S} \models_{\textbf{E-S}} {\sim}\mathsf{K}\, l \\
\mathcal{S} \models_{\textbf{E-S}} \mathsf{K}\, l & \text{iff} & \mathcal{S} \models_{\textbf{E-S}} {\sim}\mathsf{M}\, \mathit{not}\, l \\
\mathcal{S} \models_{\textbf{E-S}} \mathsf{K}\, \mathit{not}\, l & \text{iff} & \mathcal{S} \models_{\textbf{E-S}} {\sim}\mathsf{M}\, l.
\end{array} \tag{1.22}
$$

Observe that the last two equivalences of (1.22) are an outcome of the first two using the fact that double $\sim$ cancels out. We note that Kahl's truth conditions are given as the same as above, except that $\sim$ is replaced by *not* wherever it appears, but not for the objective literals. His language can then be extended through the equivalence results (1.22) with $\sim$ replaced by *not*. We note that *not* can also operate globally in Kahl's approach. It is interesting to observe in general that except the usages $\mathit{not}\, l$, $\mathit{not}\, \mathit{not}\, l$, etc. *not* pretends as if classical negation. For instance, K and M are dual, i.e. $\mathsf{K}\, l$ is equivalent to $\mathit{not}\, \mathsf{M}\, \mathit{not} l$. Moreover, $\mathsf{K}\, \mathit{not}\, \mathit{not}\, l$ and $\mathsf{K}\, l$ are equivalent, so are $\mathit{not}\, \mathit{not}\, \mathsf{K}\, l$ and $\mathsf{K}\, l$. We have the same results when K and M are exchanged.

Finally, we denote by $\|\varphi\|_{\textbf{E-S}}$ the set of all E-S models of an $\mathcal{L}_{\textbf{E-S}}$ formula φ.

Remark 1.2 *Starting with extended logic programming, in* **ASP** *(particularly, in* **E-S***) we use the term 'valuation' slightly different from its original definition given in* **HT** *logic (see Subsection 1.2.2). The difference is mainly because in such types of* **LP***, having strong negation added into the language beside the 'not' operator we talk about literals as the principal building blocks (i.e., atoms) rather than propositional variables. In particular, in* **E-S** *such atoms are further extended and divided into two parts: objective literals and subjective literals. As a result, in these classes of* **LP** *valuations are regarded as subsets of Lit (or* $\mathbb{O}$*-Lit in* **E-S***) instead of simply subsets of* $\mathbb{P}$*, so we prefer calling them 'extended valuations'. To express it more formally, an extended valuation is a consistent (or coherent) subset* V *of Lit, except one case that equals Lit itself. Therefore, if a pair of complementary literals appears in an extended valuation, in other words, if it is inconsistent then it should coincide with Lit. Finally, a belief set, a simple theory and an E-S valuation have exactly the same structure, and in one sense they are all synonyms defining the answer set concept while original valuation concept is just capable of defining stable models, but not answer sets. Here are some examples particularly about E-S valuations.*

- *The E-S valuation* $V = \{p, q\}$ *gives us that* p *and* q *are true, yet different from its original counterpart, we have no idea about the truth of other variables.*
- *The E-S valuation* $V' = \{p, {\sim}q\}$ *allows us to interpret* q *as false, as well as* p *as true. Now, it is clear that in* **E-S** *falsifying a variable is given through explicitly presenting its strong negation in the valuation. However,*

the truth values of the rest is again unknown. Hence, $V' \models_{\textbf{E-S}} not\ r$, but also $V' \models_{\textbf{E-S}} not \sim r$. Moreover, $V' \models_{\textbf{E-S}} not \sim p$ and $V' \models_{\textbf{E-S}} not\ q$.

To sum up, in an E-S valuation V, both truth and falsity values are determined according to being an element of V, and $\mathbb{O}$-Lit $\setminus V$ helps us interpret unknown, in other words, the 'not' operator.

The semantics of **E-S** is given by the notion of *world view*. Intuitively, it is a collection of maximally-generated simple theories none of which is contained in some other element, and each simple theory describes a possible world which can be established through the instructions obtained from an epistemic specification. The formal definition is given as follows: let T be a ground epistemic specification and $\mathcal{S}$ be a non-empty collection of sets of ground objective literals that have appeared in T. For each rule $\rho \in T$, the *reduct* of ρ with respect to $\mathcal{S}$, noted $\rho^{\mathcal{S}}$, is obtained from ρ by eliminating K and M in the following steps:[1]

1. first, we transform all subjective literals containing M operator in $\rho \in T$ into the equivalent subjective literal forms containing K operator using † and ††;
2. then, we remove ρ if it contains a subjective literal g such that $\mathcal{S} \not\models_{\textbf{E-S}} g$;
3. else if ρ contains a subjective literal g such that $\mathcal{S} \models_{\textbf{E-S}} g$,
   (a) we remove from ρ all occurrences of subjective literals of the form $\sim\mathsf{K}\, l$ and $\sim\mathsf{K}\, not\ l$,
   (b) and finally replace the remaining occurrences of subjective literals of the form $\mathsf{K}\, l$ and $\mathsf{K}\, not\ l$ respectively by l and $not\ l$.

The reasoning with item 2 is as follows: when ρ contains $g \in \mathbb{S}$-Lit such that $\mathcal{S} \not\models_{\textbf{E-S}} g$ then $\mathcal{S} \models_{\textbf{E-S}} \rho$ whatever the rest of the rule is because if $\mathcal{S} \not\models_{\textbf{E-S}} g$ then $\mathcal{S} \not\models_{\textbf{E-S}} body(\rho)$ since $g \in body(\rho)$ and $body(\rho)$ is a conjunction of formulas, but then S satisfies ρ trivially. The reasoning behind items 3 (a) and 3 (b) is given respectively by $\top \wedge \varphi \leftrightarrow \varphi$ and $\mathsf{K}\, \varphi \rightarrow \varphi$.

Then the disjunctive logic program $T^{\mathcal{S}}$, which is called the *reduct* of T with respect to $\mathcal{S}$, is defined as follows:

$$T^{\mathcal{S}} = \{\rho^{\mathcal{S}} \ : \ \rho \in T\}.$$

As a result, a *world view* of T is a collection $\mathcal{S}$ of simple theories such that $\mathcal{S}$ equals the set of all answer sets of the reduct $T^{\mathcal{S}}$. When the corresponding reduct $T^{\mathcal{S}}$ has no answer sets then S fails to be a world view. For instance, $\{\emptyset\}$ is not a world

[1] The first step is correct because we have seen above that $\mathsf{M}\, l$ and $\sim\mathsf{K}\, not\ l$ are equal by definition, so are $\mathsf{M}\, not\ l$ and $\sim\mathsf{K}\, l$ (see † and ††).

view of the one rule program $T = \{p \leftarrow \mathsf{K}\ not\ p\}$ since $T^{\{\emptyset\}} = \{p \leftarrow not\ p\}$ has no answer sets. Moreover $T^{\{\{p\}\}} = T^{\{\emptyset,\{p\}\}} = \emptyset$, so they both have a unique answer set $\emptyset$. However, it neither matches with $\{p\}$ nor could cover the set $\{\emptyset, \{p\}\}$. Finally, note that $T^{\mathcal{S}} = T$ for every $\mathcal{S}$ when no epistemic operators occur in T. Then, the world view of T is unique, if any and coincides to the collection of all answer sets of T. Here are some examples.

- $T_1 = \left\{ \begin{array}{l} p \leftarrow \\ r \leftarrow p\ ,\ q \\ s \leftarrow p\ ,\ not\ q \end{array} \right.$ has one answer set $\{p, s\}$. Note that $\{p, q, r\}$ is not an answer set of T_1. So, it has the unique world view $\Big\{\{p, s\}\Big\}$.
- $T_2 = \left\{ \begin{array}{l} \sim p\ or\ q \leftarrow \\ r \leftarrow not\ q \end{array} \right.$ has one world view $\mathcal{S} = \Big\{\{\sim p, r\}, \{q\}\Big\}$ which includes all answer sets of T_2.
- However, there is one exception: $T_3 = \left\{ \begin{array}{l} r \leftarrow \\ p \leftarrow not\ q \\ q \leftarrow not\ p \\ \sim r \leftarrow not\ p \end{array} \right.$ has two answer sets $\{p, r\}$ and $\{q, r, \sim r\}$, yet the world view of T_3 just includes $\{p, r\}$, i.e., it is $\Big\{\{p, r\}\Big\}$ due to the fact that when a world view includes an inconsistent set it should only be in the form $\mathbb{O}$-*Lit*, but a world view cannot include two answer sets in which one is included in other. Therefore, the only acceptable inconsistent world view is $\{\mathbb{O}\text{-}Lit\}$, yet it is not the case in this example.

The world view definition can alternatively be given by the following fixed point equation:

$$\mathcal{S} = \{W\ :\ W \models_{\mathbf{E\text{-}S}} T^{\mathcal{S}} \text{ and } W' \not\models_{\mathbf{E\text{-}S}} T^{\mathcal{S}} \text{ for every } W' \subset W\} \quad (\star)$$

Note that the semantics definition of **E-S** is given by a fixed point operation rather than declaratively and this operation is supervised by a minimisation procedure. We call the elements of a world view *belief set* (or equivalently, the *answer sets* of the corresponding reduct). Note that a belief set is always consistent, except one case when the world view equals $\{\mathbb{O}\text{-}Lit\}$. Here are some simple examples on world views of some programs.

- An epistemic specification may have finitely many (including none) world views.
  - $T_1 = \{p \leftarrow \sim\mathsf{K}\, p\}$ and $T_1' = \{p \leftarrow \sim\mathsf{M}\, p\}$ have no world views.
  - $T_2 = \{p \leftarrow \sim\mathsf{M}\, q\}$ and $T_2' = \{p \leftarrow \sim\mathsf{K}\, q\}$ have 1 world view: $\mathcal{S} = \{\{p\}\}$.

– $T_3 = \left\{ \begin{array}{l} p \leftarrow \\ q \ or \ {\sim}r \leftarrow \\ s \leftarrow \mathsf{K}\,{\sim}p \\ {\sim}t \leftarrow \mathsf{M}\,q \end{array} \right.$ has 1 world view: $\mathcal{S} = \{\{p, q, {\sim}t\}, \{p, {\sim}r, {\sim}t\}\}$.

– $T_4 = \left\{ \begin{array}{l} p \leftarrow \\ q \ or \ s \leftarrow \mathsf{K}\,p \end{array} \right.$ has 1 world view: $\mathcal{S} = \{\{p, q\}, \{p, s\}\}$.

– $T_5 = \{p \leftarrow \mathsf{M}\,p\}$ has 2 world views: $\mathcal{S}_1 = \{\emptyset\}$ and $\mathcal{S}_2 = \{\{p\}\}$.

– $T_6 = \left\{ \begin{array}{l} p \leftarrow {\sim}\mathsf{M}\,q \\ q \leftarrow {\sim}\mathsf{M}\,p \end{array} \right.$ and $T_7 = \left\{ \begin{array}{l} p \leftarrow {\sim}\mathsf{K}\,q \\ q \leftarrow {\sim}\mathsf{K}\,p \end{array} \right.$ both have 2 world views: $\mathcal{S}_1 = \{\{p\}\}$ and $\mathcal{S}_2 = \{\{q\}\}$.

Kahl's reduct definition just differs in item 2 above: when $\mathcal{S} \not\models_{\textbf{E-S}} g$ where $g = {\sim}\mathsf{K}\,l$ or $g = {\sim}\mathsf{K}\ not\,l$, he respectively replaces g by $not\,l$ and $not\,not\,l$. Kahl considers the following reasoning: for instance, when $\mathcal{S} \not\models_{\textbf{E-S}} {\sim}\mathsf{K}\ not\,l$ that further means $\mathcal{S} \models_{\textbf{E-S}} \mathsf{K}\ not\,l$, item 3 (b) forces to replace $\mathsf{K}\ not\,l$ by $not\,l$, so the original literal ${\sim}\mathsf{K}\ not\,l$ needs to be replaced by $not\ not\,l$ (Lifschitz et al. [1999]). Remember that the nested rules $\varphi \leftarrow \psi$, $not\ not\,\chi$ and $\varphi\ or\ not\,\chi \leftarrow \psi$ are strongly equivalent, and it may facilitate finding the answer sets of the nested reduct. As a result of the subtle difference, while Gelfond gets the world views $\{\emptyset\}$ and $\{\{p\}\}$ for

$$T_1 = \left\{ \ p \leftarrow \mathsf{M}\,p \ \right\} \tag{1.23}$$

Kahl obtains just the latter. Similarly,

$$T_2 = \left\{ \ p\ or\ q \leftarrow \ , \ \ p \leftarrow \mathsf{M}\,q \ \right\} \tag{1.24}$$

has no world views according to Gelfond's approach while Kahl's semantics has a unique world view $\{\{p\}\}$, which we find more intuitive.

Given an epistemic specification T, we call a world view of T *consistent* if all its belief sets are consistent. Moreover, we call T itself *consistent* if it has at least one consistent world view. Otherwise, it is *inconsistent*. Hence, all epistemic specifications without a world view are regarded as inconsistent. For example, the one rule program $T_1 = \{p \leftarrow \mathsf{M}\ not\ p\}$ (or equivalently $T_1' = \{p \leftarrow {\sim}\mathsf{K}\,p\}$) is inconsistent since it has no world view. Specifications with a unique world view $\{\mathbb{O}\text{-}Lit\}$[1] are another type of inconsistent epistemic specifications, and we call them specifically *contradictory*. However, sometimes a consistent epistemic program may also have $\{\mathbb{O}\text{-}Lit\}$ as a world view. For example, $T_2 = \left\{ \begin{array}{l} p \leftarrow \mathsf{M}\,p \\ {\sim}p \leftarrow \mathsf{K}\,p \end{array} \right.$

[1]There is exactly one inconsistent world view and an inconsistent belief set. As expected, the belief set is $\mathbb{O}$-*Lit*, and the world view is the singleton of this set. This fact can be easily derived from ($\star$). Thus, if a world view contains $\mathbb{O}$-*Lit* or $\emptyset$ then it cannot include any other belief sets.

is a consistent program because it has two world views, namely $\mathcal{S}_1 = \{\emptyset\}$ and $\mathcal{S}_2 = \{\mathbb{O}\text{-}Lit\}$, and the former is consistent. One should note that being *non-contradictory* does not guarantee the existence of world views because the class of noncontradictory programs also comprises the class of programs that do not have world views. Finally, it is important to see that an epistemic specification cannot have two world views such that one is (strictly) included in the other because a world view is maximally generated by having contained all answer sets of the corresponding reduct. Moreover, a world view cannot contain two different belief sets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathcal{S}_1 \subset \mathcal{S}_2$ because they are the answer sets of a reduct which is basically a disjunctive logic program and we know that no disjunctive logic program can have two such answer sets ([Gelfond and Lifschitz, 1991, Lemma 1]). To sum up, a world view of a program, if it exists, is either consistent or equals $\{\mathbb{O}\text{-}Lit\}$.

Finally in this subsection, we list a bunch of fundamental examples by Gelfond which shows at the same time the historical improvements of epistemic specifications. The first is a well-known motivating example by Gelfond (Gelfond [2011]): we first consider a set of rules among which are used to award scholarships to the students in a certain college:

1. $eligible(X) \leftarrow highGPA(X)$
2. $eligible(X) \leftarrow fairGPA(X)\ ,\ minority(X)$
3. $\sim eligible(X) \leftarrow \sim fairGPA(X)\ ,\ \sim highGPA(X)$
4. $interview(X) \leftarrow not\ eligible(X)\ ,\ not\ \sim eligible(X)$
5. $interview(X) \leftarrow \sim \mathsf{K}\, eligible(X)\ ,\ \sim \mathsf{K} \sim eligible(X)$

where X ranges over a given set of students. The first three rules are self-contained. However, the fourth and the fifth rules need information that is obtained from the first three rules. On the other hand, although the last two rules seem to refer to the same statement, in fact the fourth rule is not powerful enough to formalise the intended statement, that is to say,
$(*)$ "*The students whose eligibility is not determined by the college rules should be interviewed by the scholarship committee*"
The reason is because the *not* operator works (locally) in each belief set separately, and does not concern with the global situation. However, the intuitive meaning of the rule five has proved to naturally correspond to the statement $(*)$ because K operates (globally) through the belief sets altogether, so is able to capture a common knowledge in the existence of multiple belief sets. Now, let us justify this fact through a specific example: in addition to the above-mentioned program

rules, for instance, if we regard a database (DB) consisting of the following three facts

6. $fairGPA(ann) \leftarrow$
7. $\sim highGPA(ann) \leftarrow$
8. $fairGPA(mike)$ *or* $highGPA(mike) \leftarrow$

the program $T_1 = \{1-4, 6, 7\}$ consisting of the rules 1 - 4, 6 and 7 has exactly one answer set, and so the unique world view:

$$\mathcal{S} = \Big\{\{fairGPA(ann), \sim highGPA(ann), interview(ann)\}\Big\}.$$

However, when we replace the rule 4 in T_1 by the rule 5, the resulting theory T_2 receives the same world view as T_1 because $T_2^{\mathcal{S}} = \{1-3, interview(ann) \leftarrow, 6, 7\}$ since $\mathcal{S} \models_{\text{E-S}} \sim\mathsf{K}\, eligible(ann)$ and $\mathcal{S} \models_{\text{E-S}} \sim\mathsf{K} \sim eligible(ann)$, and $T_2^{\mathcal{S}}$ has the same answer set as T_1. One reason for this result is because we do not have a disjunctive rule in the list of our database we considered for T_1 and T_2, so we get exactly one belief set for both programs. Thus, we conclude that in the existence of a single belief set, the epistemic operators K and M as well as the negation as failure operator *not* coincide functionally. The diversity significantly appears when the disjunctive rules like 8 are added into a program. For example, if we consider the program $T_3 = \{1-4, 8\}$ then the resulting answer sets are as follows:

$$S_1 = \{highGPA(mike), eligible(mike)\}$$
$$S_2 = \{fairGPA(mike), interview(mike)\}$$

Then, it is straightforward that T_3 has exactly one world view, i.e., $\mathcal{S}' = \{S_1, S_2\}$. Hence, the reasoner associated with T_3 will answer *unknown* to the questions of eligible(mike) and interview(mike) for the scholarship. To spell it out, although Mike's eligibility for the scholarship is not determined by the college rules, we cannot say Mike should be sent to the interview. As a result, the rule 4 fails to represent the statement $(*)$. However, when we replace the rule 4 in T_3 by the rule 5, we see that the intended behaviour of the new system T_4 results in saying again *unknown* to the query of Mike's eligibility for the scholarship, but *yes* to the question of interview(mike). So, now the functioning of the system exactly suits $(*)$. Formally speaking, the resulting theory T_4 has the following world view $\mathcal{S}'' = \{S_3, S_4\}$ where

$$S_3 = \{fairGPA(mike), interview(mike)\} \text{ and}$$
$$S_4 = \{highGPA(mike), eligible(mike), interview(mike)\}.$$

Following the reduct definition, we get $T_4^{\mathcal{S}''} = \{1-3, interview(mike) \leftarrow, 8\}$ since $\mathcal{S}'' \models_{\text{E-S}} \sim\mathsf{K}\, elligible(mike)$ and $\mathcal{S}'' \models_{\text{E-S}} \sim\mathsf{K} \sim elligible(mike)$ and the rest is routine. To sum up, this example justifies the need for epistemic operators in $\mathcal{L}_{\text{E-S}}$.

The following example points at the requirement of a slight improvement in the old definition of world views because as it was first recognized by Teodor Przimusinski, under the old definition the world views may sometimes reflect some unintended behaviours for the system (Gelfond [2011]). In this work we do not include the old definition of world views, but one can refer to Gelfond [1991, 1994, 2011] for a detailed information. Now we consider the epistemic specification $T = \{p \leftarrow \mathsf{K}\, p\}$. According to the new definition, T has the unique world view $\mathcal{S} = \{\emptyset\}$ which conforms with the intended behaviour of the system. However, the old definition also produces an unsupported belief as contained in $\mathcal{S}' = \Big\{\{p\}\Big\}$ and it is clearly not expected by the rational agent. Therefore, Gelfond has to redesign the language and the semantics of epistemic specifications. Even if there is no ultimate success with all problems that have appeared under the old definition, the new version attains successfully to eliminate, at least, some of the unintended interpretations. Therefore, it seems that the new semantics substantially meets the requirements of intelligent agents which are capable of introspective reasoning with incomplete information. Moreover, it looks to better fit to the intuitive meanings of the epistemic operators K and M as well.

Epistemic specifications can also be used as an alternative formalisation of the closed world assumption (CWA) (Gelfond [2011]; Reiter [1978]). The CWA briefly says that $p(X)$ should be assumed to be false if there is no evidence to the contrary and is normally expressed by an ASP rule: $\sim p(X) \leftarrow \ not\ p(X)$. However, it has an alternative representation specifically as an E-S rule: $\sim p(X) \leftarrow \mathsf{K}\ not\ p(X)$ (Gelfond [2011]).

Finally, we recall the computational property of epistemic specifications: deciding whether an epistemic specification has a world view is PSPACE-complete (Zhang [2006]).

1.5 Relating ASP to other nonmonotonic formalisms

The two research areas, **LP** and nonmonotonic reasoning, progressed largely independently of each other until a new declarative semantics is proposed for logic programs. Since then people have discovered their close relationship, and logic programs have been shown to be equivalent to suitable forms of four major nonmonotonic formalisms: McCarthy's circumscription (McCarthy [1980, 1986]), Reiter's closed world assumption (Reiter [1978]), Moore's autoepistemic logic (Moore

[1985b, 1988]) and Reiter's default logic (Reiter [1980]). The importance of these results stems not only from the fact that they shed new light on the relationship between logic programming and non-monotonic reasoning, but also from the fact that they establish a close relationship between four major formalisations of non-monotonic reasoning for an important class of theories.

The **ASP** programs have close relations with nonmonotonic formalisms (Przymusinski [1988b]) such as autoepistemic logic, default logic, etc. The stable model definition can be equivalently given by means of reducing logic programs to a fixed point nonmonotonic formalism such as autoepistemic logic, default logic or introspective circumscription. For instance, extended logic programs can be seen as autoepistemic theories (Lifschitz and Schwarz [1993]). Some other relations between **ASP** and nonmonotonic formalisms are given below.

1.5.1 Default logic and ASP

General logic programs can be seen as default theories (see Appendix C.1) in the sense of Reiter (Reiter [1980]). Similarly, the language of extended nondisjunctive **LP** can be embedded into default logic: extended logic programs are identical to a natural, easily identifiable subset of default logic. The backward arrow $\leftarrow$ in **ASP** is similar to the backward arrow in the default logic context. The role of *not* in **ASP** is similar to the role of justifications in a default theory. Intuitively, default logic can express facts "p is true by default" and $not\, p$ means "p is false by default". An extension for a default theory is a theory in the sense of classical logic. Accordingly, the symbol $\sim$ in **ASP** is classical negation in default logic.

The embedding from extended nondisjunctive **LP** to default logic is explained in Gelfond et al. [1991] in detail. Below we just give its outline. We first transform an extended logic program rule r (see Definition 1.6)

$$l_1 \leftarrow (l_2\,,\ \ldots\ ,\ l_n)\,,\ (not\, l_{n+1}\,,\ \ldots\ ,\ not\, l_k)$$

into the corresponding default d_r (see Definition C.1 in Appendix C.1)

$$l_1 \leftarrow (l_2 \wedge \ldots \wedge l_n)\ :\ \mathrm{M}\overline{l_{n+1}}, \ldots, \mathrm{M}\overline{l_k} \tag{1.25}$$

where $\overline{l_i}$ stands for the complementary literal of l_i for $1 \leq i \leq k$ (for the definition of *complementary*, see Subsection 1.1.1, second paragraph). Hence, every extended logic program Π can be transformed into the corresponding default theory D_Π where

$$D_\Pi = \{d_r\ :\ r \in \Pi\}.$$

To express it more clearly, every extended logic program rule r can be viewed as a default $d_r \in D_\Pi$, in which the consequent and the justifications are simply literals and the prerequisite is a conjunction of literals.

We now define a function Γ_D with respect to a default theory D:

$$\Gamma_D \ : \ E \mapsto \Gamma_D(E)$$

mapping a set of sentences E to the smallest set of sentences $\Gamma_D(E)$ satisfying:

1. for any $d \in D$, if $G \in \Gamma_D(E)$ and $\neg H_i \notin E$ for any i then $F \in \Gamma_D(E)$ and
2. $\Gamma_D(E)$ is deductively closed.

The latter means that this set contains all sentences that can be deduced from itself. We call E an extension for D if $\Gamma_D(E) = E$.

Then we capture the answer set semantics of extended **LP** as follows:

Proposition 1.2 *For an extended program* Π*,*

1. *if* S *is an answer set of* Π *then the deductive closure of* S *is an extension of* Π*, and*
2. *every extension of* Π *is the deductive closure of exactly one answer set of* Π*.*

To sum up, the deductive closure operator is a 1-1 mapping between the answer sets of Π and its extensions.

However, this embedding cannot be generalised easily to disjunctive programs. The main difficulty is because there is a difference between epistemic disjunction (*or*) and classical disjunction ($\vee$) respectively used in extended **LP** and default logic. The reader can refer Subsection 1.1.2.2 where a similar difference has already been discussed. Following example illustrates this difference briefly.

- The disjunctive program $\Pi = \left\{ \ p \, or \, q \leftarrow \right.$ has two answer sets, namely $\{p\}$ and $\{q\}$. On the other hand, the default theory $D = \left\{ \ p \vee q \leftarrow \ : \right.$ has one extension, i.e., the deductive closure of $p \vee q$.

1.5.2 Use of the CWA in ASP

The following extended program rules in the form of predicates represent the CWA

$$\sim P(x) \leftarrow not \ P(x)$$
$$P(x) \leftarrow not \sim P(x)$$

and whenever an explicit declaration is required they can be added into any program to make definitions of predicates complete. The example below can be found in Gelfond and Lifschitz [1991]. We first consider a program Π containing the rule which specifies "any employed person has an adequate income":

1. $Adequate{-}Income(x) \leftarrow Employed(x, y)$

and the database consisting of the employment of two people:

2. $Employed(Jack, Stanford) \leftarrow$

3. $Employed(Jane, Harvard) \leftarrow$.

The answer set of Π is:

$$\{2, 3, Adequate{-}Income(Jack), Adequate{-}Income(Jane)\}.$$

However, the employment status of both Jack and Jane is not complete because, for instance, we do not know if Jack is also employed at Harvard or not. To make it more precise we can allow for the CWA for *Employed*:

4. $\sim Employed(x, y) \leftarrow not\ Employed(x, y)$.

Having rule 4 added into Π, the resulting program Π' allows us to append the literals

$$\sim Employed(Jack, Harvard) \quad \text{and} \quad \sim Employed(Jane, Stanford)$$

to the previous answer set. If we want to be strict just for the employment in the Stanford University, then we can use:

5. $\sim Employed(x, Standford) \leftarrow not\ Employed(x, Stanford)$.

instead of rule 4, and in this case, the resulting program Π'' will just add the literal

$$\sim Employed(Jane, Standford)$$

into the former answer set.

1.6 Structure of the dissertation: our work in a nutshell

The dissertation consists of six chapters. The present chapter, i.e., Introduction constitutes a background for a further study about the foundations of **ASP** and equilibrium logic. The rest of this dissertation is organised in a way that chapters 2, 3 and 4 constitute the backbone of our work. The remaining chapters conclude the dissertation with some final remarks and research goals.

Chapter 2 begins with a brief recall of previously proposed modal formalisms underlying equilibrium logic (Fariñas del Cerro and Herzig [2011a]) and explains

the drawbacks of those approaches. Then, it introduces a newly-designed monotonic modal logic called **MEM** that is powerful enough to characterise the existence of an equilibrium model as well as the consequence relation in equilibrium models. Moreover, **MEM** seems to overcome all syntactical deficiencies of the previous approaches. It has two modal operators [T] and [S] that are interpreted in a fairly standard class of Kripke models. We relate the language of equilibrium logic to our bimodal language by means of the Gödel translation tr whose main clause is:

$$\mathsf{tr}(\varphi \to \psi) = [\mathrm{T}](\mathsf{tr}(\varphi) \supset \mathsf{tr}(\psi)).$$

The logic **MEM** thus captures the minimisation attitude that is central in the definition of equilibrium models and which is only formulated in the metalanguage. Moreover, it encodes the existence of an equilibrium model of a formula, and of equilibrium consequence problem in equilibrium logic.

Chapter 3 reports on a first approach on the extensions of equilibrium logic with modal concepts. It adds an analysis of the dynamics by integrating operators of upgrading and downgrading propositional variables. More explicitly, we extend the language of **HT** logic by two kinds of atomic programs allowing to minimally update the truth value of a propositional variable here or there, if possible. These atomic programs are then combined by the usual dynamic logic program connectives. Briefly, we define a simple logic called dynamic here-and-there logic (**D-HT**) of atomic change of equilibrium models and discover that it is strongly related to dynamic logic of propositional assignments (**DL-PA**): propositional assignments set the truth values of propositional variables to either true or false and update the current model in the style of dynamic epistemic logics. Eventually, we come up with another monotonic modal logic underlying equilibrium logic.

Chapter 4 once again tackles the problem of trying to find out a more comprehensive framework of **ASP** and presents an alternative approach: it extends the original language of **HT** logic by adding modal operators of knowledge, and the resulting epistemic extension of equilibrium logic proposes a new logical semantics to Gelfond's epistemic specifications—the extensions of **ASP** by epistemic modal operators in order to correctly represent incomplete information in situations when there are multiple answer sets. As a result, this chapter paves the way from epistemic specifications to epistemic **ASP**, and can be regarded as a nice starting point for further frameworks of extensions of **ASP**. Moreover, we also provide a strong equivalence result for EHT theories (finite sets of EHT formulas).

Finally we conclude with Chapter 5 giving a brief summary of this dissertation and pointing out number of directions for future research.

Chapter 2

Capturing Equilibrium Models in Modal Logic: MEM

HT models and equilibrium models were investigated so as to characterise **ASP**. The semantics of equilibrium logic is given in an indirect way: the notion of equilibrium model is defined in terms of satisfiability in propositional logic and in the logic of here-and-there (**HT**).

We here give a direct semantics of equilibrium logic in terms of a modal language extending that of propositional logic by two unary modal operators, [T] and [S]. Roughly speaking, [T] allows to talk about valuations that are at least as strong as the actual valuation; and [S] allows to talk about valuations that are weaker than the actual valuation. Our modal language can be interpreted on HT models. However, we also give a semantics in terms of Kripke models. We call our logic **MEM**: the *modal logic of equilibrium models*. Then, we also give a sound and complete axiomatisation of our logic. Moreover, we show that it can be checked whether $\chi \mid\!\approx \varphi$, i.e., whether φ is a logical consequence of χ in equilibrium logic, by checking if the modal formula

$$\Big(\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)\Big) \supset \mathsf{tr}(\varphi)$$

is valid in **MEM** or not, where tr is a polynomial translation from the language of **HT** logic ($\mathcal{L}_{\mathbf{HT}}$) into the language of **MEM**.

Capturing one formalism in terms of another is a natural issue for theoretical discourse, however capturing equilibrium logic in a modal logic have not been discussed in detail so far. A first attempt to relate equilibrium logic to modal logic in the style of the present approach was presented in Fariñas del Cerro and Herzig [2011a]. In this chapter, we extend and improve that work by simplifying the translation. What we do for equilibrium logic in this chapter parallels what Levesque did for autoepistemic logic. Likewise, he also designed a monotonic modal

logic that was able to capture nonmonotonic autoepistemic reasoning (Lakemeyer and Levesque [2012]; Levesque [1990]).

The chapter is organised as follows. In Section 2.1 we introduce our modal logic of equilibrium models **MEM**[1] syntactically, semantically and also axiomatically. In Section 2.2 we define the Gödel translation tr from the language of **HT** logic to the language of **MEM** and prove its correctness: for every formula φ, φ is HT valid if and only if $\mathsf{tr}(\varphi)$ is **MEM** valid. This theorem paves the way for the proof of the grand finale given in the same section: φ is a logical consequence of χ in equilibrium logic if and only if the modal formula

$$\Big(\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)\Big) \supset \mathsf{tr}(\varphi)$$

is valid in **MEM**. It follows that φ has an equilibrium model if and only if $\mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$ is satisfiable in the corresponding Kripke model. Section 2.3 mentions some further research avenues related with **MEM**.

2.1 The modal logic of equilibrium models: MEM

We introduce the modal logic of equilibrium models **MEM** in the classical way: we start by defining its bimodal language and its semantics. Then we axiomatise its validities.

2.1.1 Language ($\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$)

Throughout the chapter we suppose $\mathbb{P}$ is given a countably infinite set of propositional variables. The elements of $\mathbb{P}$ are noted p, q, etc. Our language $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ is bimodal: it has two modal operators, namely [T] and [S]. Precisely, $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ is defined by the following grammar:

$$\varphi ::= p \mid \bot \mid \varphi \supset \varphi \mid [\mathrm{T}]\varphi \mid [\mathrm{S}]\varphi,$$

where p ranges over $\mathbb{P}$. The formula $[\mathrm{T}]\varphi$ may be read "φ holds at every possible there-world at least as strong as the current world", and $[\mathrm{S}]\varphi$ may be read "φ holds at every possible weaker or equal here-world".

The set of propositional variables occurring in a formula φ is noted $\mathbb{P}_\varphi$.

The language $\mathcal{L}_{[\mathrm{T}]}$ is the set of $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formulas without the modal operator [S]. So the $\mathcal{L}_{[\mathrm{T}]}$-formulas are built from [T] and the Boolean connectives only.

[1] To avoid confusion we could have used another name instead of **MEM** again. It should however be clear to the reader that the modal logic we are talking about here is just slightly different from the one that is introduced in Fariñas del Cerro and Herzig [2011a].

We use the following standard abbreviations: $\top \stackrel{\mathsf{def}}{=} \bot \supset \bot$, $\sim\varphi \stackrel{\mathsf{def}}{=} \varphi \supset \bot$, $\varphi \vee \psi \stackrel{\mathsf{def}}{=} \sim\varphi \supset \psi$, $\varphi \wedge \psi \stackrel{\mathsf{def}}{=} \sim(\varphi \supset \sim\psi)$, and $\varphi \equiv \psi \stackrel{\mathsf{def}}{=} (\varphi \supset \psi) \wedge (\psi \supset \varphi)$. Moreover, $\langle \mathrm{T} \rangle \varphi$ and $\langle \mathrm{S} \rangle \varphi$ respectively abbreviate $\sim[\mathrm{T}]\sim\varphi$ and $\sim[\mathrm{S}]\sim\varphi$.

2.1.2 MEM frames

We interpret the formulas of our language $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ in a class of Kripke models that has to satisfy the below-mentioned particular constraints. In this subsection, we first give these (first-order) constraints together with the special names attributed to them. Then, we clarify their meaning with some comments. Finally, we come up with an axiomatisation and also prove its completeness.

Consider the class of Kripke frames $(W, \mathcal{T}, \mathcal{S})$ such that

- W is a non-empty set of possible worlds;
- $\mathcal{T}, \mathcal{S} \subseteq W \times W$ are (binary) relations on W such that:

$\mathrm{refl}(\mathcal{T})$	for every $w, w\mathcal{T}w$;
$\mathrm{alt}_2(\mathcal{T})$	for every w, u, u', u'', if $w\mathcal{T}u$, $w\mathcal{T}u'$ and $w\mathcal{T}u''$ then $u = u'$ or $u = u''$ or $u' = u''$;
$\mathrm{trans}(\mathcal{T})$	for every w, u, v, if $w\mathcal{T}u$ and $u\mathcal{T}v$ then $w\mathcal{T}v$;
$\mathrm{refl}_2(\mathcal{S})$	for every w, u, if $w\mathcal{S}u$ then $u\mathcal{S}u$;
$\mathrm{wtriv}_2(\mathcal{S})$	for every w, u, v, if $w\mathcal{S}u$ and $u\mathcal{S}v$ then $u = v$;
$\mathrm{wmconv}(\mathcal{T}, \mathcal{S})$	for every w, u, if $w\mathcal{T}u$ then $w = u$ or $u\mathcal{S}w$;
$\mathrm{mconv}(\mathcal{S}, \mathcal{T})$	for every w, u, if $w\mathcal{S}u$ then $u\mathcal{T}w$.

We call a frame $(W, \mathcal{T}, \mathcal{S})$ satisfying the above-mentioned constraints a **MEM** frame. Let us explain these constraints informally.

To begin with, the first three constraints are about the relation $\mathcal{T}$. The constraints $\mathrm{refl}(\mathcal{T})$ and $\mathrm{alt}_2(\mathcal{T})$ say respectively that a world w is $\mathcal{T}$-reflexive and has at most two $\mathcal{T}$-successors. To sum it up, a world w is either a single $\mathcal{T}$-loop or has an accompanying $\mathcal{T}$-accessible world. Then the transitivity constraint, $\mathrm{trans}(\mathcal{T})$, makes that the neighbouring $\mathcal{T}$-accessible world is a single $\mathcal{T}$-loop. Briefly, these constraints together imply the following constraint about the relation $\mathcal{T}$:

$$\mathrm{depth}_1(\mathcal{T}): \quad \text{for every } w, u, v, \text{ if } w\mathcal{T}u \text{ and } u\mathcal{T}v \text{ then } w = u \text{ or } u = v.$$

In words, every world can be reached in at most one $\mathcal{T}$-step.

The next two constraints are about the relation $\mathcal{S}$. Let $\mathcal{S}(u) = \{v : u\mathcal{S}v\}$. For any w, u if $w\mathcal{S}u$ the constraint $\mathrm{refl}_2(\mathcal{S})$ gives us $u \in \mathcal{S}(u)$. The constraint $\mathrm{wtriv}_2(\mathcal{S})$ tells us that when $w\mathcal{S}u$ then we must have $\mathcal{S}(u) = \emptyset$ or $\mathcal{S}(u) = \{u\}$. Together, they

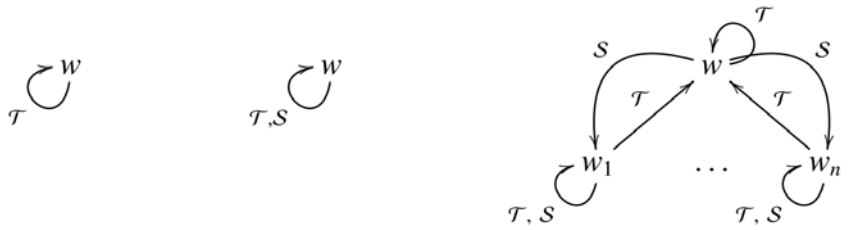

Figure 2.1: Graphical representation of **MEM**-frames, for $n \geq 0$. The two singleton graphs are for $n = 0$. The rightmost graph is for $n \geq 1$, where $f(n) = 2n$ represents the number of $\mathcal{S}$-arrows in the diagram.

say that if $w\mathcal{S}u$ then $\mathcal{S}(u) = \{u\}$: any world we access by the relation $\mathcal{S}$ can see itself through $\mathcal{S}$, but none of the others. At this point, it is worth noting that $\mathcal{S}$ is trivially transitive due to $\mathrm{wtriv}_2(\mathcal{S})$. It then also follows from this constraint that every world can be reached in at most one $\mathcal{S}$-step. In other words, the relation $\mathcal{S}$ is of depth 1.

The next two constraints involve both $\mathcal{T}$ and $\mathcal{S}$. We obtain from the weak mixed conversion constraint, $\mathrm{wmconv}(\mathcal{T}, \mathcal{S})$, that $\mathcal{T}$ is contained in $\mathcal{S}^{-1} \cup \Delta_W$, where $\Delta_W = \{(w, w) \ : \ w \in W\}$ is the diagonal of $W \times W$. Moreover, the mixed conversion constraint, $\mathrm{mconv}(\mathcal{S}, \mathcal{T})$, says that $\mathcal{S}$ is contained in $\mathcal{T}^{-1}$. As a result, together with $\mathrm{refl}(\mathcal{T})$ these two constraints give us $\mathcal{T} = \mathcal{S}^{-1} \cup \Delta_W$.

Let us sum up the constraints that we have introduced so far: the $\mathcal{T}$ relation is a tree of height 0 or 1, and $\mathcal{S}$ is the converse of $\mathcal{T}$, except for the root. In our frames, any root w is characterized by the fact that $\mathcal{T}(w) \setminus \{w\}$ is empty. Visually, **MEM** frames are disjoint unions of the diagrams depicted by a representative in Figure 2.1. The constraints $\mathrm{wmconv}(\mathcal{T}, \mathcal{S})$, $\mathrm{refl}_2(\mathcal{S})$, $\mathrm{wtriv}_2(\mathcal{S})$, $\mathrm{refl}(\mathcal{T})$ and $\mathrm{alt}_2(\mathcal{T})$ imply that for every w, $\mathcal{T}(w) \cap \mathcal{S}(w)$ is equal to either the empty-set or the singleton $\{w\}$[1].

The following properties hold for every **MEM** frame $(W, \mathcal{T}, \mathcal{S})$. First, the relation $\mathcal{T}$ is serial, i.e., for all w there is a u such that $w\mathcal{T}u$. Formally, this property is guaranteed by the constraint $\mathrm{refl}(\mathcal{T})$. Moreover, $\mathcal{T}$ is directed, i.e., for every w, u, v, if $w\mathcal{T}u$ and $w\mathcal{T}v$ then there exists z such that $u\mathcal{T}z$ and $v\mathcal{T}z$. This follows from the constraints $\mathrm{refl}(\mathcal{T})$ and $\mathrm{alt}_2(\mathcal{T})$. Besides, $\mathcal{T}$ is also anti-symmetric, that is to say, for every w, u, if $w\mathcal{T}u$ and $u\mathcal{T}w$ then $w = u$. This follows from the constraints $\mathrm{wmconv}(\mathcal{T}, \mathcal{S})$ and $\mathrm{wtriv}_2(\mathcal{S})$. Together with $\mathrm{mconv}(\mathcal{S}, \mathcal{T})$, this implies that $\mathcal{S}$ is anti-symmetric, too. However, $\mathcal{T}$ is not euclidean: we may have $w\mathcal{T}u$ and $w\mathcal{T}w$ without $u\mathcal{T}w$, and therefore the condition 'for every w, u, v, if $w\mathcal{T}u$ and $w\mathcal{T}v$

[1] correct the graph.

then $u\mathcal{T}v$' does not hold in general. Finally, the relations $\mathcal{T}$ and $\mathcal{S}$ are trivially idempotent.[2] We obtain the idempotence property of $\mathcal{T}$ from $\text{depth}_1(\mathcal{T})$, while we get that of $\mathcal{S}$ through $\text{wtriv}_2(\mathcal{S})$. As a last word, all of the properties above can be visualized from the diagram above; in addition, we can also see that the properties of seriality, euclideanity, and directedness don't hold for the relation $\mathcal{S}$.

2.1.3 MEM models

We interpret the formulas of our language $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ in a class of Kripke models that has to satisfy some particular constraints.

Consider the class of Kripke models $M = (W, \mathcal{T}, \mathcal{S}, V)$ such that:

- $(W, \mathcal{T}, \mathcal{S})$ is a **MEM** frame;
- V is a valuation on W mapping all possible worlds $w \in W$ to sets of propositional variables $V_w \subseteq \mathbb{P}$ such that:

heredity($\mathcal{S}$)	for every w, u, if $w\mathcal{S}u$ then $V_u \subseteq V_w$;
neg($\mathcal{S}, \mathcal{T}$)	for every w, there exists u such that: $w\mathcal{T}u$ and for every $P \subset V_u$, there is v satisfying $u\mathcal{S}v$ and $V_v = P$.

A quadruple $M = (W, \mathcal{T}, \mathcal{S}, V)$ satisfying all the conditions above is called a **MEM** model.

Now let us comment a bit on the constraints heredity($\mathcal{S}$) and neg($\mathcal{S}, \mathcal{T}$). They involve not only the relations $\mathcal{S}$ and $\mathcal{T}$, but also the valuation V. The constraint heredity($\mathcal{S}$) is just as the heredity constraint of intuitionistic logic, except that $\mathcal{S}$ is the inverse of the intuitionistic relation. The neg($\mathcal{S}, \mathcal{T}$) constraint basically says that the set of worlds that are accessible from a root u of a tree with non-empty valuation via the relation $\mathcal{S}$ contains all those worlds v whose valuations V_v are strictly included in V_u. In every **MEM** model, if singleton (isolated) points appear (as in the leftmost two graphs in Figure 2.1) then they should certainly have an empty valuation.

The following properties include the valuation as well.

Proposition 2.1 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model.*

1. *For every w, u, if $w\mathcal{T}u$ then $V_w \subseteq V_u$.*
2. *For every w, if $\mathcal{T}(w) \setminus \{w\}$ is empty, then the set $\{V_u : w\mathcal{S}u\}$ equals either $\{V : V \subseteq V_w\}$ or $\{V : V \subset V_w\}$.*

[2] A relation r is idempotent if $r \circ r = r$, where $\circ$ is a relational composition.

PROOF. See Proposition A.1 and its proof in Section A.2 of Appendix A. q.e.d.

While the first property above gives the heredity property of intuitionistic logic for $\mathcal{T}$, the second says, in words: for a root point w,[1] the set of valuations of the worlds that are accessible from w via $\mathcal{S}$ is either the set of subsets of V_w (2^{V_w}) or the set of strict subsets of V_w ($2^{V_w} \setminus \{V_w\}$). This property will be used later, exactly in the proof of Theorem 3.4.

2.1.4 Truth conditions

The semantics of our bimodal logic is fairly standard, the relation $\mathcal{T}$ interpreting the modal operator [T] and the relation $\mathcal{S}$ interpreting the modal operator [S]. The truth conditions are:

$$
\begin{array}{lll}
M, w \models p & \text{iff} & p \in V_w; \\
M, w \not\models \bot; & & \\
M, w \models \varphi \supset \psi & \text{iff} & M, w \not\models \varphi \text{ or } M, w \models \psi; \\
M, w \models [\mathrm{T}]\varphi & \text{iff} & M, u \models \varphi \text{ for every } u \text{ such that } w\mathcal{T}u; \\
M, w \models [\mathrm{S}]\varphi & \text{iff} & M, u \models \varphi \text{ for every } u \text{ such that } w\mathcal{S}u.
\end{array}
$$

We say that φ has a **MEM** model when $M, w \models \varphi$ for some model M and world w in M. We also say that φ is **MEM** *satisfiable*. Furthermore, φ is **MEM** *valid* if and only if $M, w \models \varphi$ for every model M and possible world w in M.

The next proposition says that to check satisfiability it suffices just to consider models with finite valuations.

Proposition 2.2 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model, and φ be an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula. Let the valuation V^φ be defined as follows:*

$$V^\varphi_w = V_w \cap \mathbb{P}_\varphi, \text{ for every } w \in W.$$

Then $M^\varphi = (W, \mathcal{T}, \mathcal{S}, V^\varphi)$ is also a **MEM** *model. Moreover, for every $w \in W$,*

$$M, w \models \varphi \text{ if and only if } M^\chi, w \models \varphi,$$

where φ is a subformula of χ.

PROOF. See Proposition A.2 and its proof in Section A.2 of Appendix A. q.e.d.

Remark 2.1 *Observe that Proposition 2.2 should not be confused with the finite model property (f.m.p.) of modal logics: the f.m.p. is about finiteness of the set of possible worlds, while Proposition 2.2 is about finiteness of valuations. We might call the latter finite valuation property (f.v.p.).*

[1] Remember that in a **MEM** frame, the property $\mathcal{T}(w) \setminus \{w\} = \emptyset$ characterises that w is the root of a tree. Moreover, when $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ then the singleton $\mathcal{T}(w) \setminus \{w\}$ contains the root.

2.1.5 Axiomatics, provability, and completeness

The main purpose of this subsection is to give an axiomatisation of the **MEM** validities and to prove its completeness.

To start with, we define the fragment of *positive Boolean formulas* of $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ by the following grammar:

$$\varphi^+ ::= p \mid \varphi^+ \wedge \varphi^+ \mid \varphi^+ \vee \varphi^+.$$

We immediately observe that every positive Boolean formula is falsifiable. (Note that this holds because $\top$ is not a positive Boolean formula.)

Now we are ready to give our axiomatisation of **MEM**. The axiom schemas and the inference rules are listed in Table 2.1.

$\mathbf{K}([\mathrm{T}])$	the axioms and the inference rules of modal logic **K** for [T]
$\mathbf{K}([\mathrm{S}])$	the axioms and the inference rules of modal logic **K** for [S]
$\mathrm{T}([\mathrm{T}])$	$[\mathrm{T}]\varphi \supset \varphi$
$\mathrm{Alt}_2([\mathrm{T}])$	$[\mathrm{T}]\varphi \vee [\mathrm{T}](\varphi \supset \psi) \vee [\mathrm{T}](\varphi \wedge \psi \supset \bot)$
$4([\mathrm{T}])$	$[\mathrm{T}]\varphi \supset [\mathrm{T}][\mathrm{T}]\varphi$
$\mathrm{T}_2([\mathrm{S}])$	$[\mathrm{S}]([\mathrm{S}]\varphi \supset \varphi)$
$\mathrm{WTriv}_2([\mathrm{S}])$	$[\mathrm{S}](\varphi \supset [\mathrm{S}]\varphi)$
$\mathrm{WMConv}([\mathrm{T}],[\mathrm{S}])$	$\varphi \supset [\mathrm{T}](\varphi \vee \langle \mathrm{S}\rangle\varphi)$
$\mathrm{MConv}([\mathrm{S}],[\mathrm{T}])$	$\varphi \supset [\mathrm{S}]\langle \mathrm{T}\rangle\varphi$
$\mathrm{Heredity}([\mathrm{S}])$	$\langle \mathrm{S}\rangle\varphi^+ \supset \varphi^+$ for φ^+ a positive Boolean formula
$\mathrm{Neg}([\mathrm{S}],[\mathrm{T}])$	$\langle \mathrm{T}\rangle(\varphi^+ \wedge \psi) \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle(\sim\varphi^+ \wedge \psi)$ for φ^+ a positive Boolean formula such that $\mathbb{P}_{\varphi^+} \cap \mathbb{P}_{\psi} = \emptyset$

Table 2.1: Axiomatisation of **MEM**

The axiom schemas $\mathrm{T}([\mathrm{T}])$, $\mathrm{Alt}_2([\mathrm{T}])$ and $4([\mathrm{T}])$ are well-known from modal logic textbooks. We observe that $\mathrm{Alt}_2([\mathrm{T}])$ could be replaced by the axiom schema $\langle \mathrm{T}\rangle(\varphi \wedge \psi) \wedge \langle \mathrm{T}\rangle(\varphi \wedge \sim\psi) \supset [\mathrm{T}]\varphi$, or even $(\varphi \wedge \psi) \wedge \langle \mathrm{T}\rangle(\varphi \wedge \sim\psi) \supset [\mathrm{T}]\varphi$.

The axiom schema $\mathrm{WTriv}_2([\mathrm{S}])$ is a weakening of the triviality axiom for [S], i.e., $[\mathrm{S}]\varphi \equiv \varphi$, yet the axiom schemas $\mathrm{T}_2([\mathrm{S}])$ and $\mathrm{WTriv}_2([\mathrm{S}])$ can be combined into the single axiom, $[\mathrm{S}]([\mathrm{S}]\varphi \equiv \varphi)$ that we call 'triviality in the second step' axiom, and symbolise with $\mathrm{Triv}_2([\mathrm{S}])$.

The weak mixed conversion axiom $\mathrm{WMConv}([\mathrm{T}],[\mathrm{S}])$ and the mixed conversion axiom $\mathrm{MConv}([\mathrm{S}],[\mathrm{T}])$ are familiar from tense logics.

Finally, the schema Heredity([S]) captures the heredity constraint heredity($\mathcal{S}$). Note that it could be replaced by the axiom $\langle S\rangle p \supset p$, where p is a propositional variable. It could also be replaced by the axiom schema $\sim\varphi^+ \supset [S]\sim\varphi^+$, for φ^+ a positive Boolean formula. The schema Neg([S], [T]) ensures that the modal operator [S] quantifies over all strict subsets of the actual valuation under some restrictions.

Remark 2.2 *T*([T]) *and* Alt_2([T]) *give us that for every* w, $\mathcal{T}(w)$ *contains* w, *and has at least 1 and at most 2 elements. If* $\mathcal{T}(w)$ *contains two elements, say* $\mathcal{T}(w) = \{w, u\}$ *where* $w{\neq}u$, *then* 4([T]) *implies that* $\mathcal{T}(u) = \{u\}$. *Moreover,* *WMConv*([T], [S]) *guarantees that* u *is always* $\mathcal{S}$*-related to* w *when it exists.*

Finally, the notions of *proof* and of *provability of a formula* are defined as it is in any modal logic. For example, it is possible to prove the schema Depth_1([T]) (corresponding to the constraint $\text{depth}_1(\mathcal{T})$), i.e., $\sim\varphi \supset [T](\varphi \supset [T]\varphi)$. The proof uses the axiom schemas T([T]), Alt_2([T]), and 4([T]).

As another example, we give the proof of the schema that corresponds to the heredity condition for $\mathcal{T}$, i.e., Heredity([T]): $\varphi^+ \supset [T]\varphi^+$, for φ^+ a positive Boolean formula. This will be helpful in the proof of the grand finale, Theorem 3.4.

Proposition 2.3 *The schema Heredity*([T]), *i.e.,* $\varphi^+ \supset [T]\varphi^+$, *for* φ^+ *a positive Boolean formula, is provable.*

Proof. See Proposition A.3 and its proof in Section A.2 of Appendix A. q.e.d.

Heredity([T]) ensures that $w\mathcal{T}u$ implies $V_w \subseteq V_u$, i.e., the heredity condition for $\mathcal{T}$. Here is one more schema just concerning the $\mathcal{T}$ relation.

Proposition 2.4 *The schema* 2([T]), *i.e.,* $\langle T\rangle[T]\varphi \supset [T]\langle T\rangle\varphi$ *is provable.*

Proof. See Proposition A.4 and its proof in Section A.2 of Appendix A. q.e.d.

Let us turn to schemas about [S]. For example, 4([S]): $[S]\varphi \supset [S][S]\varphi$ is a direct consequence of WTriv_2([S]). The proof is in one step by the K([S]) axiom and modus ponens (MP).

Finally, we state and prove a schema regarding both operators [T] and [S] that will be useful in the completeness proof.

Lemma 2.1 *The following formula schema is provable:*

$$Neg'([S],[T]) \qquad \langle T\rangle\Big((\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \langle T\rangle\langle S\rangle\Big((\bigwedge_{p\in P} \sim p) \wedge (\bigwedge_{q\in Q} q)\Big)$$

for $P, Q \subseteq \mathbb{P}$ *finite,* $P \neq \emptyset$, *and* $P \cap Q = \emptyset$.

PROOF. See Lemma A.2 and its proof in Section A.2 of Appendix A. q.e.d.

Our axiomatisation is sound and complete.

Theorem 2.1 *Let φ be an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula. Then φ is* **MEM** *valid if and only if φ is provable from the axioms and the inference rules of* **MEM**.

PROOF. See Theorem A.2 and its proof in Section A.2 of Appendix A. q.e.d.

2.2 Embedding HT logic and equilibrium logic into the modal logic MEM

In this section we are going to translate HT logic and equilibrium logic into our logic **MEM**.

2.2.1 Translating $\mathcal{L}_{\mathbf{HT}}$ into $\mathcal{L}_{[\mathrm{T}]}$

To warm up, let us translate the language $\mathcal{L}_{\mathbf{HT}}$ of both HT logic and equilibrium logic into the sub-language $\mathcal{L}_{[\mathrm{T}]}$ of **MEM**. We recursively define the mapping tr as follows:

$$\begin{array}{lcll} \mathsf{tr}(p) & = & p & \text{for } p \in \mathbb{P}; \\ \mathsf{tr}(\bot) & = & \bot; & \\ \mathsf{tr}(\varphi \wedge \psi) & = & \mathsf{tr}(\varphi) \wedge \mathsf{tr}(\psi); & \\ \mathsf{tr}(\varphi \vee \psi) & = & \mathsf{tr}(\varphi) \vee \mathsf{tr}(\psi); & \\ \mathsf{tr}(\varphi \rightarrow \psi) & = & [\mathrm{T}](\mathsf{tr}(\varphi) \supset \mathsf{tr}(\psi)). & \end{array}$$

This translation is similar to the Gödel translation from intuitionistic logic to modal logic **S4** whose main clause is $\mathsf{tr}(\varphi \rightarrow \psi) = \Box(\mathsf{tr}(\varphi) \supset \mathsf{tr}(\psi))$, where $\Box$ is an **S4** operator (just as the [T] operator of our bimodal logic). Here are some examples.

$$\mathsf{tr}(\top) = \mathsf{tr}(\bot \rightarrow \bot) = [\mathrm{T}](\bot \supset \bot).$$

This is equivalent to $\top$ in any normal modal logic.

$$\mathsf{tr}(\neg p) = \mathsf{tr}(p \rightarrow \bot) = [\mathrm{T}](p \supset \bot).$$

This is equivalent to $[\mathrm{T}]{\sim}p$ in any normal modal logic.

$$\mathsf{tr}(p \vee \neg p) = \mathsf{tr}(p) \vee \mathsf{tr}(p \rightarrow \bot) = p \vee [\mathrm{T}](p \supset \bot).$$

This is equivalent to $p \vee [\mathrm{T}]{\sim}p$ in any normal modal logic.

2.2.2 Correspondence between HT logic and MEM

In this subsection, we will see that the fragment $\mathcal{L}_{[T]}$ of the language $\mathcal{L}_{[T],[S]}$ is at least as expressive as $\mathcal{L}_{\mathbf{HT}}$ on HT models, modulo the translation tr. First of all, given a set of propositional variables $T \subseteq \mathbb{P}$, we will construct a special **MEM** model, M_T, and will check the satisfiability in HT logic over satisfiability in **MEM** via this special **MEM** model, M_T. Briefly, using these kinds of models, we will prove that our translation preserves the satisfiability.

Proposition 2.5 *For $T \subseteq \mathbb{P}$, let $M_T = (W, \mathcal{T}, \mathcal{S}, V)$ be a Kripke model such that:*

$$\begin{array}{lcl} W & = & 2^T; \\ V_H & = & H, \quad \text{for every } H \in W; \\ \mathcal{T} & = & \Delta_W \cup (W \times \{T\}) = \{(x, y) \in W \times W \ : \ x = y \text{ or } y = T\}; \\ \mathcal{S} & = & \Delta_{(W \setminus \{T\})} \cup \Big(\{T\} \times (W \setminus \{T\})\Big) \ = \ \mathcal{T}^{-1} \setminus \{(T, T)\}. \end{array}$$

Then M_T is a **MEM** *model. Furthermore, for every $\mathcal{L}_{\mathbf{HT}}$-formula φ and $H \subseteq T$,*

$$H, T \models \varphi \text{ if and only if } M_T, H \models \mathsf{tr}(\varphi).$$

Proof. See Theorem A.5 and its proof in Section A.2 of Appendix A. q.e.d.

In the proposition above, we define $\mathcal{S}$ as the relative complement of $\{(T, T)\}$ in $\mathcal{T}^{-1}$. To clarify the model, we give some examples: for $T = \emptyset$ we obtain $M_\emptyset = (W, \mathcal{T}, \mathcal{S}, V)$ with $W = \{\emptyset\}$, $\mathcal{T} = \{(\emptyset, \emptyset)\}$, and $\mathcal{S} = \emptyset$; and for $T = \{p\}$ we obtain $M_{\{p\}} = (W, \mathcal{T}, \mathcal{S}, V)$ with $W = \{\emptyset, \{p\}\}$, $\mathcal{T} = \{(\emptyset, \emptyset), (\emptyset, \{p\}), (\{p\}, \{p\})\}$, and $\mathcal{S} = \{(\emptyset, \emptyset), (\{p\}, \emptyset)\}$.

As a second step, we will see that our translation also reflects the satisfiability the other way around.

Proposition 2.6 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model. Then for every $w \in W$ and every $\mathcal{L}_{\mathbf{HT}}$-formula φ we have:*

1. *If $\mathcal{T}(w) \setminus \{w\} = \emptyset$ then $M, w \models \mathsf{tr}(\varphi)$ if and only if $V_w, V_w \models \varphi$;*
2. *If $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ then $M, w \models \mathsf{tr}(\varphi)$ if and only if $V_w, V_u \models \varphi$ for the uniquely determined $u \in \mathcal{T}(w) \setminus \{w\}$.*

Proof. See Proposition A.6 and its proof in Section A.2 of Appendix A. q.e.d.

As a result, now we can say that the satisfiability is invariant under our translation because by propositions 2.5 and 2.6, we have: for every $\varphi \in \mathcal{L}_{\mathbf{HT}}$,

φ has an HT model if and only if $\mathsf{tr}(\varphi)$ is **MEM** satisfiable.

After we give the ultimate goal of this subsection, we will also prove the assertion above as a corollary of it, but first we would like to remark upon an observation about Proposition 2.6.

Remark 2.3 *One should notice that Proposition 2.6 can actually be given as a corollary of Proposition 2.5: to see this, we take an arbitrary* **MEM** *model, say* $M = (W, \mathcal{T}, \mathcal{S}, V)$*, and a point* $w \in W$*. Then, according to the type of* w*, that is, according to* w *being the root of a tree in* M *or not (see Remark 2.2), we decide the generator of our* **MEM** *model and construct it as it is described in Proposition 2.5. So, now let us go over two cases :*
Case (i): let w *be the root of a tree in* M *then* $\mathcal{T}(w) = \{w\}$*, i.e.,* $\mathcal{T}(w) \setminus \{w\} = \emptyset$*. Therefore, we take* $T = V_w$*, and construct* M_{V_w} *as it is described in Proposition 2.5. Again by Proposition 2.5, we know that* M_{V_w} *is a* **MEM** *model, and furthermore, that* $V_w, V_w \models \mathsf{tr}(\varphi)$ *if and only if* $M_{V_w}, V_w \models \mathsf{tr}(\varphi)$*, for every* $\mathcal{L}_{\mathbf{HT}}$*-formula* φ*. Since* $\mathsf{tr}(\varphi)$ *includes neither* $[\mathrm{S}]$ *nor* $\langle \mathrm{S} \rangle$*, we assert that* $M_{V_w}, V_w \models \mathsf{tr}(\varphi)$ *if and only if* $M^*, V_w \models \mathsf{tr}(\varphi)$ *(*$\star$*) where* M^* *is a weakly generated submodel of* M_{V_w}*, i.e., just w.r.t.* $\mathcal{T}$ *in which the domain is restricted to* $\{V_w\}$ *(for generated submodel definition, see Blackburn et al. [2001a]). In the meanwhile, we ignore* $\mathcal{S}$*, that is to say, we keep it as* $\mathcal{S} = \emptyset$*. (the resulting model* M^* *is not a* **MEM** *model anymore, but since we use it to relate the main models it doesn't violate the proof) The claim (*$\star$*) is proved by the well-known fact that modal satisfaction is invariant under generated submodels. On the other hand, when we change the domain of* M^* *with* $\{w\}$ *then we obtain a weakly generated submodel of* M *(again only w.r.t.* $\mathcal{T}$*), and without subtlety the resulting model is equivalent to* M^**. Then, it clearly follows that* $V_w, V_w \models \mathsf{tr}(\varphi)$ *if and only if* $M_{V_w}, V_w \models \mathsf{tr}(\varphi)$ *if and only if* $M^*, V_w \models \mathsf{tr}(\varphi)$ *if and only if* $M, w \models \mathsf{tr}(\varphi)$*, for every* $\mathcal{L}_{\mathbf{HT}}$*-formula* φ*.*
Case (ii): let w *be a leaf of a tree structure in* M *then* $\mathcal{T}(w) = \{w, u\}$ *for a uniquely determined* $u \neq w$*. Thus, we get* $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$*. Here we take* $T = V_u$*, and form the* **MEM** *model* M_{V_u} *again as in Proposition 2.5. Therefore, Proposition 2.5 gives us the following:* $V_w, V_u \models \mathsf{tr}(\varphi)$ *if and only if* $M_{V_u}, V_w \models \mathsf{tr}(\varphi)$*, for every* $\mathcal{L}_{\mathbf{HT}}$*-formula* φ*. The rest of the proof follows basically the same as case (i), so we leave it to the reader.*

Now we are about to prove the correctness of our translation. To spell it out, we are ready to embed the notion of HT validity into **MEM**.

Theorem 2.2 *Given an* $\mathcal{L}_{\mathbf{HT}}$*-formula,* φ*,*

$$\varphi \textit{ is HT valid if and only if } \mathsf{tr}(\varphi) \textit{ is } \mathbf{MEM} \textit{ valid.}$$

Proof. See Proposition A.2 and its proof in Section A.2 of Appendix A. q.e.d.

After we have seen that the validity is invariant under our translation tr now the corollary below immediately follows.

Corollary 2.1 *For every $\mathcal{L}_{\mathbf{HT}}$-formula φ,*

$$\varphi \textit{ has an HT model if and only if } \mathsf{tr}(\varphi) \textit{ is } \mathbf{MEM} \textit{ satisfiable.}$$

Proof. See Proposition A.1 and its proof in Section A.2 of Appendix A. q.e.d.

2.2.3 Correspondence between equilibrium logic and MEM

The same construction as for HT logic allows us to turn equilibrium models into **MEM** models.

Proposition 2.7 *Given $T \subseteq \mathbb{P}$, let $M_T = (W, \mathcal{T}, \mathcal{S}, V)$ be a Kripke model such that:*

$$\begin{array}{lcl} W & = & 2^T; \\ V_H & = & H, \quad \textit{for every } H \in W; \\ \mathcal{T} & = & \Delta_W \cup (W \times \{T\}); \\ \mathcal{S} & = & \Delta_{(W \setminus \{T\})} \cup \Big(\{T\} \times (W \setminus \{T\})\Big). \end{array}$$

Then M_T is a **MEM** *model, and T is an equilibrium model of φ if and only if $M_T, T \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$, for every $\mathcal{L}_{\mathbf{HT}}$-formula φ.*

Proof. See Proposition A.7 and its proof in Section A.2 of Appendix A. q.e.d.

To clarify the proposition, we consider an example: take $T = \emptyset$ and $\varphi = \top$. We have seen before that $\emptyset$ is the only equilibrium model of $\top$ (for verification, see the second paragraph of Introduction). Let $M_\emptyset$ be the **MEM** model as constructed in propositions 2.5 and 2.7 (for the description of the model, see the paragraph just before the proof of Proposition 2.5). By the same propositions, we know that $M_\emptyset$ is a legal **MEM** model. We also deduce $M_\emptyset, \emptyset \models \mathsf{tr}(\top) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\top)$ following the conclusion of Proposition 2.7 and the structure of the model. This can also be seen by simplifying the latter:

$$\begin{array}{lll} \mathsf{tr}(\top) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\top) & \text{iff} & \top \wedge [\mathrm{S}]{\sim}\top \\ & \text{iff} & [\mathrm{S}]\bot. \end{array}$$

Proposition 2.7 helps us give an alternative proof of the result mentioned in Remark 1.1 in Chapter 1: to see this, let $T \subseteq \mathbb{P}$ be an equilibrium model of an $\mathcal{L}_{\mathbf{HT}}$-formula φ. Then, we construct the **MEM** model M_T as in Proposition 2.7, and we conclude that $M_T, T \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$. Hence, we get $M_T, T \models \mathsf{tr}(\varphi)$ and $M_T, H \not\models \mathsf{tr}(\varphi)$ for every $H \subset T$. From this observation we obtain that $T \setminus \mathbb{P}_\varphi$

is empty; otherwise $M_T, (T \cap \mathbb{P}_\varphi) \cup Q \models \mathsf{tr}(\varphi)$ for every $Q \subseteq T \setminus \mathbb{P}_\varphi$ (note that $\mathbb{P}_\varphi = \mathbb{P}_{\mathsf{tr}(\varphi)}$), and since $(T \cap \mathbb{P}_\varphi) \cup Q \subset T$ for every $Q \subset T \setminus \mathbb{P}_\varphi$ that would bring about a contradiction. Consequently, if T is an equilibrium model of $\varphi \in \mathcal{L}_{\mathbf{HT}}$ then $T \subseteq \mathbb{P}_\varphi$.

Proposition 2.8 *Given a* **MEM** *model* $M = (W, \mathcal{T}, \mathcal{S}, V)$ *and* $w \in W$, *if* $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ *then let* $u \in \mathcal{T}(w) \setminus \{w\}$, *or else let* $u = w$. *Then*

1. *if* $V_u = \emptyset$ *then* $M, u \models \mathsf{tr}(\varphi)$ *if and only if* V_u *is an equilibrium model for* φ,
2. *and if* $V_u \neq \emptyset$ *then* $M, u \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$ *if and only if* V_u *is an equilibrium model for* φ, *for every* $\mathcal{L}_{\mathbf{HT}}$*-formula* φ.

Proof. See Proposition A.8 and its proof in Section A.2 of Appendix A. q.e.d.

Given Remark 1.1 mentioned in Chapter 1, one can get confused of Proposition 2.8 since he/she can think that V_u is not necessarily a subset of $\mathbb{P}_\varphi$, and that it is not finite either. However, one should keep in mind that the condition '$M, u \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$' forces V_u to be equal to $V_u \cap \mathbb{P}_\varphi$. We cannot use the same condition '$M, u \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$' for the first item in which $V_u = \emptyset$, and we don't need either because for $T = \emptyset$ the minimality condition in the definition of equilibrium model trivially holds, and the condition '$M, u \models \mathsf{tr}(\varphi)$' guarantees the rest of the definition. To see the claim, we give a simple counterexample: take a **MEM** model $M = (W, \mathcal{T}, \mathcal{S}, V)$ in which for $u \in W$ such that $\mathcal{T}(u) = \{u\}$, V_u is empty and an $\mathcal{L}_{\mathbf{HT}}$-formula $\varphi = \neg p$. We know that empty-set is the (only) equilibrium model for $\neg p$. However, '$M, u \models \mathsf{tr}(\neg p) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\neg p)$' doesn't hold in general (the only case it trivially holds is the singleton point w with $\mathcal{S}(w) = \emptyset$, and in all other cases it fails).

We are now ready for the grand finale where we embed the notion of logical consequence in equilibrium logic into our **MEM** logic, so we will end the line of work that aims to capture equilibrium logic in our bimodal logic, **MEM**.

Theorem 2.3 *Given* $\mathcal{L}_{\mathbf{HT}}$*-formulas* χ *and* φ,

$$\chi \mid\!\approx \varphi \text{ if and only if } \Big(\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)\Big) \supset \mathsf{tr}(\varphi) \text{ is } \mathbf{MEM} \text{ valid.}$$

Proof. See Theorem A.5 and its proof in Section A.2 of Appendix A. q.e.d.

Corollary 2.2 *For every* $\mathcal{L}_{\mathbf{HT}}$*-formula* χ,

$$\chi \text{ has an equilibrium model iff } \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi) \text{ is } \mathbf{MEM} \text{ satisfiable.}$$

PROOF. See Corollary A.2 and its proof in Section A.2 of Appendix A. q.e.d.

Here is an example. We have seen before that $\top \mid\!\approx \neg p$ for every p, i.e., $\neg p$ is a consequence of $\top$ in equilibrium models. Hence Theorem 3.4 tells us that the formula $\xi = \mathsf{tr}(\top) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\top) \supset \mathsf{tr}(\neg p)$ must be provable from the axioms and the inference rules of **MEM**. This can be established by the following sequence of equivalent formulas. Before, we recall that in any normal modal logic, $\mathsf{tr}(\top)$ is equivalent to $\top$ and $\mathsf{tr}(\neg p)$ is equivalent to $[\mathrm{T}]{\sim}p$ (see Section 3.3.3).

1. $\mathsf{tr}(\top) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\top) \supset \mathsf{tr}(\neg p)$
2. $\top \wedge [\mathrm{S}]{\sim}\top \supset [\mathrm{T}]{\sim}p$ (see above)
3. $[\mathrm{S}]\bot \supset [\mathrm{T}]{\sim}p$.

The last line is provable in our logic: indeed, we see below that $[\mathrm{S}]\bot \supset [\mathrm{T}]{\sim}p$ can be proved in our logic **MEM** by standard principles of modal logic.

1. $\langle \mathrm{T}\rangle(p \wedge \top) \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle({\sim}p \wedge \top)$ (axiom Neg($[\mathrm{S}], [\mathrm{T}]$))
2. $\langle \mathrm{T}\rangle p \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle{\sim}p$ (from 1 by classical logic)
3. ${\sim}p \supset \top$ (tautology)
4. $\langle \mathrm{T}\rangle\langle \mathrm{S}\rangle{\sim}p \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (from 3 by **K**($[\mathrm{T}]$) and **K**($[\mathrm{S}]$))
5. $\langle \mathrm{T}\rangle p \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (from 2 and 4)
6. $p \supset \langle \mathrm{T}\rangle p$ (axiom T ($[\mathrm{T}]$))
7. $p \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (from 5 and 6)
8. $\langle \mathrm{T}\rangle p \supset \langle \mathrm{T}\rangle\langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (from 7 by **K**($[\mathrm{T}]$))
9. $\langle \mathrm{T}\rangle\langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (axiom 4 ($[\mathrm{T}]$))
10. $\langle \mathrm{T}\rangle p \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\top$ (from 8 and 9)
11. $\langle \mathrm{T}\rangle(\langle \mathrm{S}\rangle\top \wedge [\mathrm{S}]\langle \mathrm{S}\rangle\top) \supset \langle \mathrm{S}\rangle\top$ (axiom WMConv($[\mathrm{T}], [\mathrm{S}]$))
12. $[\mathrm{S}](\top \supset \langle \mathrm{S}\rangle\top)$ (axiom T_2($[\mathrm{S}]$))
13. $[\mathrm{S}](\langle \mathrm{S}\rangle\top \supset \top)$ (axiom WTriv_2($[\mathrm{S}]$))
14. $[\mathrm{S}]\langle \mathrm{S}\rangle\top \equiv [\mathrm{S}]\top$ (from 12 and 13 by **K**($[\mathrm{S}]$))
15. $[\mathrm{S}]\langle \mathrm{S}\rangle\top \equiv \top$ (from 14 by **K**($[\mathrm{S}]$))

16. $\langle \mathrm{T} \rangle \langle \mathrm{S} \rangle \top \supset \langle \mathrm{S} \rangle \top$ (from 11 and 15 by $\mathbf{K}([\mathrm{T}])$ and $\mathbf{K}([\mathrm{S}])$)

17. $\langle \mathrm{T} \rangle p \supset \langle \mathrm{S} \rangle \top$ (from 10 and 16)

18. $[\mathrm{S}] \bot \supset [\mathrm{T}] {\sim} p$ (from 17 by $\mathbf{K}([\mathrm{T}])$ and $\mathbf{K}([\mathrm{S}])$).

Therefore the original formula ξ is also provable in our logic.

2.3 Conclusion and future work

Besides embedding a nonmonotonic logic into a monotonic logic, our logic **MEM** has a further interesting feature that may be exploited in a future work: we can now apply well-known automated deduction methods for modal logics (Enjalbert and Fariñas del Cerro [1989]; Fariñas del Cerro and Herzig [1995]; Hustadt and Schmidt [2000]; Schmidt and Tishkovsky [2008]; Sebastiani and Tacchella [2009]) to equilibrium logic. We may use in particular our LoTREC tableau proving platform (Fariñas del Cerro et al. [2001]). The implementation of a tableau procedure for **MEM** requires a specific tableau rule that does the following: for each subset of the set of propositional variables appearing in some node, create an $\mathcal{S}$-accessible node where all these variables are false.

In the following chapter, we will see another monotonic modal logic **DL-PA** that is able to embed equilibrium logic. This result was already published in Fariñas del Cerro et al. [2013].

Chapter 3

Combining Equilibrium Logic and Dynamic Logic

In this chapter, we present two approaches: (1) we introduce an alternative monotonic modal logic underlying equilibrium logic, and also (2) we introduce an extension of equilibrium logic with dynamic modal operators.

In parallel to the extensions of the original language of **ASP**, we here propose a different, but more modest approach, where the new piece of information is restricted to be atomic. It is based on the update of HT models. We consider two kinds of basic update operations: one sets a propositional variable true either here or there according to its truth value in these sets; similarly the other sets it false either here or there, again as far as its truth value is concerned. From these basic update operations we allow to build update programs by means of the standard dynamic logic program operators of sequential and nondeterministic composition, iteration, and test. We call the resulting formalism *dynamic here-and-there logic* (**D-HT**).

The notions of an equilibrium model and of logical consequence in equilibrium models is defined exactly as before. We also recall dynamic logic of propositional assignments (**DL-PA**) that was studied in Balbiani et al. [2013], and define a translation tr from the language of **D-HT** into the language of **DL-PA**. Our main result says that a formula φ is an equilibrium consequence of a formula χ if and only if the **DL-PA** formula

$$\langle \pi_1 \rangle \Big(\mathsf{tr}(\chi) \wedge {\sim} \langle \pi_2 \rangle \mathsf{tr}(\chi) \supset \mathsf{tr}(\varphi) \Big)$$

is valid, where π_1 and π_2 are **DL-PA** programs whose length is polynomial in the length of χ and φ.

The chapter is organised as follows. In Section 3.1 we introduce dynamic here-and-there logic (**D-HT**) and define consequence in its equilibrium models. In

Section 3.2 we present dynamic logic of propositional assignments (**DL-PA**). In Section 3.3 we define translations relating the language of **D-HT** to the language of **DL-PA** and vice versa. Section 3.4 concludes the chapter giving our further interests related with **DL-PA**.

3.1 A dynamic extension of HT logic and of equilibrium logic

In this section we propose a dynamic extension of here-and-there (**HT**) logic, named **D-HT**. By means of the standard definition of an equilibrium model, this extension also provides a definition of a non-monotonic consequence relation which is a conservative extension of the standard equilibrium consequence relation.

To begin with, we fix a countably infinite set of propositional variables ($\mathbb{P}$) whose elements are noted p, q, etc. The language is produced through adding dynamic modalities to the language of **HT** logic. The semantics is based on HT models.We write $\mathbb{HT}$ for the set of all HT models: $\mathbb{HT} = \{(H, T) \ : \ H \subseteq T \subseteq \mathbb{P}\}$.

3.1.1 Language ($\mathcal{L}_{\text{D-HT}}$)

The language $\mathcal{L}_{\text{D-HT}}$ is defined by the following grammar:

$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid [\pi]\varphi \mid \langle\pi\rangle\varphi$$
$$\pi ::= +p \mid -p \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \varphi?$$

where p ranges over $\mathbb{P}$.

We have only two kinds of *atomic programs* in the language: $+p$ and $-p$. Each of them minimally updates an HT model, if this is possible: in a sense, the former 'upgrades the truth of p' while the latter 'downgrades the truth of p', again if it is possible. More precisely, the program $+p$ makes p true in there, but keeps its truth value same in here if p is not included in there. However, if p exists in there, but not in here then it makes p true in here while keeping its truth value same in there; otherwise the program $+p$ fails. On the other hand, the program $-p$ sets p false in here as it keeps it in there if p is contained in here. Nevertheless, if p is only contained in there, but not in here then the program $-p$ excludes p from there keeping its truth value same in here; or else the program fails.

Using the atomic programs described above, we can define more specific, but again primitive update operators which are symbolically expressed with $+_{\mathsf{h}}p$, $+_{\mathsf{t}}p$, $-_{\mathsf{t}}p$, and $-_{\mathsf{h}}p$:

$$+_{\mathtt{h}}p = (\langle +p\rangle[+p]\bot?\ ;+p) \cup p?$$
$$+_{\mathtt{t}}p = (\neg p?\ ;+p) \cup \neg\neg p?$$
$$-_{\mathtt{h}}p = (p?\ ;-p) \cup \langle +p\rangle\top?$$
$$-_{\mathtt{t}}p = (\langle +p\rangle[+p]\bot?\ ;-p) \cup \neg p?$$

As it is clearly seen from the statements above, the program $+_{\mathtt{t}}p$ makes p true only there, but keeps the truth value of p here. Similarly, the program $-_{\mathtt{h}}p$ makes p false just here without modifying it there. Moreover, the program $+_{\mathtt{h}}p$ makes p true here, under the condition that p is true there; else it is inexecutable. Similarly, the program $-_{\mathtt{t}}p$ sets p false there, under the condition that p is false here; else it is inexecutable.

On the other hand, it is also possible to give these expressions the other way around:

$$+p = (\neg p?\ ;+_{\mathtt{t}}p) \cup (\langle +p\rangle[+p]\bot?\ ;+_{\mathtt{h}}p)$$
$$-p = (p?\ ;-_{\mathtt{h}}p) \cup (\langle +p\rangle[+p]\bot?\ ;-_{\mathtt{t}}p).$$

So, we see that in fact they can mutually defined in terms of each other.

The operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("(.)*", the so-called Kleene star), and test ("(.)?") are familiar from propositional dynamic logic (**PDL**). We refer the reader to Appendix D for a short introduction of **PDL**.

We will generally call formulas and programs as expressions, so in this paper an *expression* is a formula or a program.

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ, with the exception of [,], $\langle$, $\rangle$, $+$, $-$, and parentheses. For example, $|p \wedge (q \vee r)| = 1 + 1 + 3 = 5$. The length of a program π, noted $|\pi|$, is defined in the same way. For example, $|\Big([+p]\bot?\ ;-p\Big)| = 3 + 1 + 1 = 5$.

For a given formula φ, the set of variables occurring in φ is noted $\mathbb{P}_{\varphi}$. For example, $\mathbb{P}_{[-p](q\vee r)} = \{p, q, r\}$.

The *static fragment* of $\mathcal{L}_{\mathbf{D\text{-}HT}}$ is the fragment of $\mathcal{L}_{\mathbf{D\text{-}HT}}$ without dynamic operators $[\pi]$ and $\langle\pi\rangle$ for every π, noted $\mathcal{L}_{\mathbf{HT}}$. This is nothing but the language of **HT** logic and of equilibrium logic.

Negation of a formula φ, noted $\neg\varphi$, is defined as the abbreviation of $\varphi \to \bot$. We also use $\top$ as a shorthand for $\bot \to \bot$.

$$
\begin{aligned}
\|p\|_{\textbf{D-HT}} &= \Big\{(H,T) \ : \ p \in H\Big\} \\
\|\bot\|_{\textbf{D-HT}} &= \emptyset \\
\|\varphi \wedge \psi\|_{\textbf{D-HT}} &= \|\varphi\|_{\textbf{D-HT}} \cap \|\psi\|_{\textbf{D-HT}} \\
\|\varphi \vee \psi\|_{\textbf{D-HT}} &= \|\varphi\|_{\textbf{D-HT}} \cup \|\psi\|_{\textbf{D-HT}} \\
\|\varphi \to \psi\|_{\textbf{D-HT}} &= \Big\{(H,T) \ : \ (H,T),(T,T) \in (\mathbb{HT} \setminus \|\varphi\|_{\textbf{D-HT}}) \cup \|\psi\|_{\textbf{D-HT}}\Big\} \\
\|[\pi]\varphi\|_{\textbf{D-HT}} &= \Big\{(H,T) \ : \ (H_1,T_1) \in \|\varphi\|_{\textbf{D-HT}} \text{ for every } ((H,T),(H_1,T_1)) \in \|\pi\|_{\textbf{D-HT}}\Big\} \\
\|\langle\pi\rangle\varphi\|_{\textbf{D-HT}} &= \Big\{(H,T) \ : \ (H_1,T_1) \in \|\varphi\|_{\textbf{D-HT}} \text{ for some } ((H,T),(H_1,T_1)) \in \|\pi\|_{\textbf{D-HT}}\Big\} \\
\|{+}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_2 \setminus H_1 = \{p\} \text{ and } T_2 = T_1, \text{ or } T_2 \setminus T_1 = \{p\} \text{ and } H_2 = H_1\Big\} \\
\|{-}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_1 \setminus H_2 = \{p\} \text{ and } T_2 = T_1, \text{ or } T_1 \setminus T_2 = \{p\} \text{ and } H_2 = H_1\Big\} \\
\|\pi_1;\pi_2\|_{\textbf{D-HT}} &= \|\pi_1\|_{\textbf{D-HT}} \circ \|\pi_2\|_{\textbf{D-HT}} \\
\|\pi_1 \cup \pi_2\|_{\textbf{D-HT}} &= \|\pi_1\|_{\textbf{D-HT}} \cup \|\pi_2\|_{\textbf{D-HT}} \\
\|\pi^*\|_{\textbf{D-HT}} &= \|\pi\|^*_{\textbf{D-HT}} \\
\|\varphi?\|_{\textbf{D-HT}} &= \Big\{((H,T),(H,T)) \ : \ (H,T) \in \|\varphi\|_{\textbf{D-HT}}\Big\}
\end{aligned}
$$

Table 3.1: Interpretation of the **D-HT** connectives.

3.1.2 Dynamic here-and-there logic: D-HT

We display below the interpretation of formulas and programs together at a time: the interpretation $\|\varphi\|_{\textbf{D-HT}}$ of a formula φ is a set of HT models, while the interpretation $\|\pi\|_{\textbf{D-HT}}$ of a program π is a relation on the set of HT models, $\mathbb{HT}$. Note that the interpretation of the dynamic connectives differs from that of usual modal logics because there is a single relation interpreting programs (that therefore does not vary with the models). The definitions are in Table 3.1.

We find it useful to also employ the interpretations of primitive update operators we mention above.

$$
\begin{aligned}
\|{+_{\mathsf{h}}}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_2 \setminus H_1 = \{p\} \text{ and } T_2 = T_1\Big\} \\
\|{+_{\mathsf{t}}}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_2 = H_1 \text{ and } T_2 \setminus T_1 = \{p\}\Big\} \\
\|{-_{\mathsf{h}}}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_1 \setminus H_2 = \{p\} \text{ and } T_2 = T_1\Big\} \\
\|{-_{\mathsf{t}}}p\|_{\textbf{D-HT}} &= \Big\{((H_1,T_1),(H_2,T_2)) \ : \ H_2 = H_1 \text{ and } T_1 \setminus T_2 = \{p\}\Big\}
\end{aligned}
$$

For instance, $\|\neg p\|_{\textbf{D-HT}}$ is the set of HT models (H,T) such that $p \notin T$ (and therefore $p \notin H$ by the heredity constraint). Hence, $\|p \vee \neg p\|_{\textbf{D-HT}}$ is the set of HT models (H,T) such that $p \in H$ or $p \notin T$. $\|\neg\neg p\|_{\textbf{D-HT}}$ is the set of HT models (H,T) such that $p \in T$. Moreover, $\|\langle{+}p\rangle\top\|_{\textbf{D-HT}}$ is the set of HT models (H,T) such that $p \notin H$: when $p \in H$ then p cannot be upgraded and the $+p$ program is inexecutable. Finally, the models of the following formula are HT-models (H,T)

where T contains p and H does not.

$$\begin{aligned}\|\langle +p\rangle\top \wedge \langle -p\rangle\top\|_{\textbf{D-HT}} &= \|\neg\neg p\|_{\textbf{D-HT}} \cap \left(\mathbb{HT} \setminus \|p\|_{\textbf{D-HT}}\right)\\ &= \|\neg\neg p\|_{\textbf{D-HT}} \cap \|\langle +p\rangle\top\|_{\textbf{D-HT}}\\ &= \{(H,T) \ : \ p \notin H \text{ and } p \in T\}\end{aligned}$$

A formula φ is **D-HT** *valid* if and only if every HT model is also a model of φ, i.e., $\|\varphi\|_{\textbf{D-HT}} = \mathbb{HT}$. For example, neither $\langle +p\rangle\top$ nor $\langle -p\rangle\top$ is valid, but $\langle +p \cup -p\rangle\top$ is. Moreover, $[+p][+p]p$, $[-p][-p]\neg p$, and $[p? \cup \neg p?](p \vee \neg p)$ are all valid. Finally, the following equivalences are valid:

$$\begin{aligned}[-p]\bot &\leftrightarrow \neg p\\ \langle -p\rangle\top &\leftrightarrow \neg\neg p\\ [+p]\bot &\leftrightarrow p\end{aligned}$$

Therefore $[-p]\bot$, $\langle -p\rangle\top$ and $[+p]\bot$ can all be expressed in $\mathcal{L}_{\textbf{HT}}$. In contrast, $\langle +p\rangle\top$ cannot because there is no formula in the static fragment $\mathcal{L}_{\textbf{HT}}$ that conveys the information such that $p \in T \setminus H$: to see this, we first take an arbitrary $p \in \mathbb{P}$, and without loss of generality, consider $\mathbb{P}$, restricted to $\mathbb{P}' = \{p\}$. Then, we assume for a contradiction that there exists an $\mathcal{L}_{\textbf{HT}}$-formula, say φ (by the assumption above, we have $\mathbb{P}_\varphi \subseteq \{p\}$), which characterizes the property of p being just in the relative difference of there in here in a given HT model, i.e., for some $\mathcal{L}_{\textbf{HT}}$-formula φ,

$$p \in T \setminus H \quad \text{if and only if} \quad (H,T) \in \|\varphi\|_{\textbf{D-HT}} \quad (\star),$$

for every HT model (H,T). Now, let us see all possible HT models over $\mathbb{P}'$, namely $(\emptyset,\emptyset)$, $(\emptyset,\{p\})$, and $(\{p\},\{p\})$. Clearly, among all, only $(\emptyset,\{p\})$ answers the requirements of being an HT model of φ (see $(\star)$ above). In other words, $(\emptyset,\{p\}) \in \|\varphi\|_{\textbf{D-HT}}$. Moreover, it is easy to see that the simplest HT formulas over $\mathbb{P}'$ are p, $\neg p$, $\neg\neg p$ (at this point, it should be clear to the reader that $\neg\neg p$ is not equivalent to p (see below)), $\bot$ and T. All others are well-formed combinations of these. On the other hand, as we have mentioned before, in HT logic, the semantics of negation is different from the semantics of the classical one in general. More explicitly, while $\|p\|_{\textbf{D-HT}} = \{(H,T) \ : \ p \in H\}$, we have $\|\neg p\|_{\textbf{D-HT}} = \{(H,T) \ : \ p \notin T\}$, $\|\neg\neg p\|_{\textbf{D-HT}} = \{(H,T) \ : \ p \in T\}$, $\|\neg\neg\neg p\|_{\textbf{D-HT}} = \|\neg p\|_{\textbf{D-HT}}$, and so on. Briefly, the intuitionistic negation of HT logic doesn't work minimally. In addition to all above, although $(\emptyset,\{p\}) \in \|\neg\neg p\|_{\textbf{D-HT}}$, its interpretation is not strong enough to satisfy the property of $p \in T \setminus H$. Therefore, $(\emptyset,\{p\})$ cannot be an HT model of φ. Thus, there cannot be a well-formed HT formula that characterises $p \in T \setminus H$. As a result, our extension of **HT** is more expressive than **HT** itself.

D-HT logic satisfies the heredity property of intuitionistic logic for atomic formulas: if (H, T) is an HT model of p then (T, T) is also an HT model of p. It is trivially satisfied because for every HT model (H, T), we have $H \subseteq T$. **D-HT** logic however fails to satisfy this property for more complex formulas containing dynamic operators. To see this, consider the HT model $(\emptyset, \{p\})$ and the formula $\langle +p \rangle \top$: $(\emptyset, \{p\})$ is a model of $\langle +p \rangle \top$, while $(\{p\}, \{p\})$ is not. However, our logic **D-HT** looks like a particular intuitionistic modal logic. Such logics were studied in the literature (Fischer-Servi [1976]). For such logics, duality of the modal operators fails as it does in **D-HT** as well: while $[\pi]\varphi \to \neg\langle\pi\rangle\neg\varphi$ is valid, the converse is invalid. For example, $(\emptyset, \emptyset)$ is an HT model of $\neg\langle +p \rangle \neg p$, but not of $[+p]p$.

It follows from the next proposition that we have a finite model property for **D-HT**: if φ has an HT model then φ has an HT model (H, T) such that T is finite.

Proposition 3.1 *Given an $\mathcal{L}_{\text{D-HT}}$-formula φ, let P be a set of propositional variables such that $P \cap \mathbb{P}_\varphi = \emptyset$. Then, for every $Q \subseteq P$,*

$$(H, T) \in \|\varphi\|_{\text{D-HT}} \quad \textit{if and only if} \quad (H \cup Q, T \cup P) \in \|\varphi\|_{\text{D-HT}}.$$

Proof. See Proposition A.9 and its proof in Section A.3 of Appendix A. q.e.d.

3.1.3 Dynamic equilibrium logic

An *equilibrium model* of an $\mathcal{L}_{\text{D-HT}}$ formula φ is a set of propositional variables $T \subseteq \mathbb{P}$ such that:

1. (T, T) is an HT model of φ;
2. no (H, T) with $H \subset T$ is an HT model of φ.

To begin with, note that (T, T) being a **D-HT** model of φ is not the same as T being a classical model of φ (where $\to$ is viewed as material implication). Therefore, we cannot replace the first condition of equilibrium model definition above by "T is a classical model of φ".

Here are some examples. The valid formulas of **D-HT** all have exactly one equilibrium model, namely the empty set. There are some formulas that have even no equilibrium model, such as $\langle -q \rangle (p \wedge q)$. This **D-HT** formula has no equilibrium model because $\langle -q \rangle (p \wedge q)$ does not even have a **D-HT** model either. The unique equilibrium model of $\langle +p \rangle (\neg p \to q)$ is $\emptyset$ because it is **D-HT** valid. Moreover, $\{p\}$ is the only equilibrium model for both $\langle -p \rangle (\neg p \to q)$, and $\langle +q; +q \rangle (p \wedge q)$.

Let χ and φ be $\mathcal{L}_{\textbf{D-HT}}$ formulas. φ is a *consequence of* χ *in equilibrium models*, written $\chi \mid\!\approx \varphi$, if and only if for every equilibrium model T of χ, (T, T) is a **D-HT** model of φ. We have discussed formulas without dynamic modalities in Chapter 1, so here are some examples involving dynamic operators. For example, $p \vee q \mid\!\approx [\neg p?]q$, and $p \vee q \mid\!\approx [\neg p?]\langle +p; +p\rangle(p \wedge q)$, but also vice versa. We also have $p \vee q \mid\!\approx [\neg p?; +p; +p](p \wedge q)$ and $[[+p]\bot? \cup [-p]\bot?](p \vee \neg p) \mid\!\approx p \vee \neg p$.

As it is clear from the definition above, in our dynamic language we can check not only problems of the form $\chi \mid\!\approx [\pi]\varphi$, but also problems of the form $\langle\pi\rangle\chi \mid\!\approx \varphi$. The former expresses a hypothetical update of χ: if χ is updated by π then φ follows. The latter may express an actual update of χ, where the program π executes the update 'the other way round': it is the converse of the original update program. For example, suppose we want to update $\chi = p \wedge q$ by $\neg q$. Updates by the latter formula can be implemented by the program $-q; -q$. Now the converse execution of $-q; -q$ is nothing but the execution of the program $\pi = +q; +q$. So, in order to know whether the update of $p \wedge q$ by $\neg q$ results in $p \wedge \neg q$ we have to check whether $\langle +q; +q\rangle(p \wedge q) \mid\!\approx p \wedge \neg q$. The latter is indeed the case: we have seen above that the only equilibrium model of $\langle +q; +q\rangle(p \wedge q)$ is $\{p\}$, and $(\{p\}, \{p\})$ is clearly a **D-HT** model of $p \wedge \neg q$. Note that the problems $\chi \vdash [\pi]\varphi$ and $\langle\pi\rangle\chi \vdash \varphi$ are deductively equivalent in **D-HT** logic because $\vdash$ is monotonic. This is not the case here.

3.2 Dynamic logic of propositional assignments: DL-PA

In this section we define syntax and semantics of dynamic logic of propositional assignments (**DL-PA**). The star-free fragment of **DL-PA** was introduced in Herzig et al. [2011], where it was shown that it embeds Coalition Logic of Propositional Control (Hoek and Wooldridge [2005]; Hoek et al. [2010]; van der Hoek and Wooldridge [2005]). The full logic with the Kleene star was further studied in Balbiani et al. [2013]. In addition to assignments of propositional variables to true or false, here we allow of assignments to arbitrary formulas as well. We need this extension in order to copy the propositional variables of a valuation and similarly, after some changes are done, to be able to retrieve the initial truth values of that valuation. We will explain these notions later in full detail. However, we keep on calling that logic **DL-PA**. This is in order because it has the same expressivity as the logic **DL-PA** of Balbiani et al. [2013].

3.2.1 Language ($\mathcal{L}_{\textbf{DL-PA}}$)

The language of **DL-PA** is defined by the following grammar:

$$
\begin{aligned}
\|p{:=}\varphi\|_{\textbf{DL-PA}} &= \{(V_1, V_2) \,:\, \text{if } V_1 \in \|\varphi\|_{\textbf{DL-PA}} \text{ then } V_2 = V_1 \cup \{p\} \text{ and if } V_1 \notin \|\varphi\|_{\textbf{DL-PA}} \text{ then } V_2 = V_1 \setminus \{p\}\} \\
\|\pi;\pi'\|_{\textbf{DL-PA}} &= \|\pi\|_{\textbf{DL-PA}} \circ \|\pi'\|_{\textbf{DL-PA}} \\
\|\pi \cup \pi'\|_{\textbf{DL-PA}} &= \|\pi\|_{\textbf{DL-PA}} \cup \|\pi'\|_{\textbf{DL-PA}} \\
\|\pi^*\|_{\textbf{DL-PA}} &= \left(\|\pi\|_{\textbf{DL-PA}}\right)^* \\
\|\varphi?\|_{\textbf{DL-PA}} &= \{(V, V) \,:\, V \in \|\varphi\|_{\textbf{DL-PA}}\} \\
\\
\|p\|_{\textbf{DL-PA}} &= \{V \,:\, p \in V\} \\
\|\bot\|_{\textbf{DL-PA}} &= \emptyset \\
\|\varphi \wedge \psi\|_{\textbf{DL-PA}} &= \|\varphi\|_{\textbf{DL-PA}} \cap \|\psi\|_{\textbf{DL-PA}} \\
\|\varphi \vee \psi\|_{\textbf{DL-PA}} &= \|\varphi\|_{\textbf{DL-PA}} \cup \|\psi\|_{\textbf{DL-PA}} \\
\|\varphi \supset \psi\|_{\textbf{DL-PA}} &= (2^{\mathbb{P}} \setminus \|\varphi\|_{\textbf{DL-PA}}) \cup \|\psi\|_{\textbf{DL-PA}} \\
\|\langle\pi\rangle\varphi\|_{\textbf{DL-PA}} &= \left\{V \,:\, \text{there is } V' \text{ such that } (V, V') \in \|\pi\|_{\textbf{DL-PA}} \text{ and } V' \in \|\varphi\|_{\textbf{DL-PA}}\right\}
\end{aligned}
$$

Table 3.2: Interpretation of the **DL-PA** connectives.

$$
\begin{array}{lcl}
\pi & ::= & p{:=}\varphi \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \varphi? \\
\varphi & ::= & p \mid \bot \mid \varphi \supset \varphi \mid \langle\pi\rangle\varphi
\end{array}
$$

where p ranges over a countably infinite set of propositional variables $\mathbb{P}$. So, an atomic program of the language of **DL-PA** is a program of the form $p{:=}\varphi$.

The *star-free fragment* of **DL-PA** is the subset of the language made up of formulas without the Kleene star "(.)*".

We abbreviate the other logical connectives in the usual way: for example, $\sim\varphi$ is defined as $\varphi \supset \bot$. In particular, $\top$ is defined as $\sim\bot \stackrel{\mathsf{def}}{=} \bot\supset\bot$. Moreover, $\varphi \vee \psi \stackrel{\mathsf{def}}{=} \sim\varphi \supset \psi$, $\varphi \wedge \psi \stackrel{\mathsf{def}}{=} \sim(\varphi \supset \sim\psi)$, and $\varphi \equiv \psi \stackrel{\mathsf{def}}{=} (\varphi \supset \psi) \wedge (\psi \supset \varphi)$. Finally, $[\pi]\varphi$ abbreviates $\sim\langle\pi\rangle\sim\varphi$, and the program `skip` abbreviates $\top?$ ("nothing happens"). Note that `skip` could also be defined by $p{:=}p$, for arbitrary p. The language of **DL-PA** allows to express the primitives of standard programming languages. For example, the loop "$\mathsf{while}\ \varphi\ \mathsf{do}\ \pi$" can be expressed as the **DL-PA** program $(\varphi?;\pi)^*;\sim\varphi?$.

3.2.2 Semantics

DL-PA programs are interpreted by means of a (unique) *relation between valuations*: atomic programs $p{:=}\varphi$ update valuations in the obvious way, and complex programs are interpreted just as in **PDL** by mutual recursion. Table 3.2 gives the interpretation of the **DL-PA** connectives.

A formula φ is **DL-PA** *valid* if $\|\varphi\|_{\textbf{DL-PA}} = 2^{\mathbb{P}}$, and it is **DL-PA** *satisfiable* if $\|\varphi\|_{\textbf{DL-PA}} \neq \emptyset$. For example, the formulas $\langle p{:=}\top\rangle\top$, $\langle p{:=}\top\rangle p$ and $\langle p{:=}\bot\rangle\sim p$ are all valid, as well as $\psi \wedge [\psi?]\varphi \supset \varphi$ and $[p{:=}\top \cup q{:=}\top](p \vee q)$. We continue with an

interesting example: (induction axiom) $\varphi \wedge [\pi^*](\varphi \supset [\pi]\varphi) \supset [\pi^*]\varphi$. Finally, if p does not occur in φ then both $\varphi \supset \langle p{:=}\top\rangle\varphi$ and $\varphi \supset \langle p{:=}\bot\rangle\varphi$ are valid. This is due to the following property that we will use while relating dynamic equilibrium logic into **DL-PA**.

Proposition 3.2 *Suppose $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables of P occurs in φ. Then $V \cup P \in \|\varphi\|_{\textbf{DL-PA}}$ iff $V \setminus P \in \|\varphi\|_{\textbf{DL-PA}}$.*

Contrarily to **PDL**, it is shown in Balbiani et al. [2013] that the Kleene star operator can be eliminated in **DL-PA**: for every **DL-PA** program π, there is an equivalent program π' such that no Kleene star occurs in π'. However, the elimination is not polynomial.

3.3 Correspondence between D-HT and DL-PA

In this section we are going to translate **D-HT** and dynamic equilibrium logic into **DL-PA**, and vice versa. These translations are polynomial and the first allows us to check **D-HT** validity and consequence in equilibrium models in **DL-PA**.

We start by defining some **DL-PA** programs that will be the building blocks in embedding some notions of **D-HT** into **DL-PA**. Some of these programs require to copy propositional variables. The key point of the translation is then to consider programs allowing to characterise a given HT model. This is done by renaming or copying the set of propositional variables appearing in the given state. Some of these programs are used in the translation.

3.3.1 Copying propositional variables

The translation from **D-HT** into **DL-PA** introduces some 'fresh' propositional variables that do not exist in the **D-HT** formula we translate. Precisely, this requires to suppose a new set of propositional variables: it is the union of the set of 'original' variables $\mathbb{P} = \{p_1, p_2, \ldots\}$ and the set of 'copies' of these variables $\mathbb{P}' = \{p'_1, p'_2, \ldots\}$, where $\mathbb{P}$ and $\mathbb{P}'$ are disjoint. The function $(.)'$ is a bijection between the powerset of these two sets: for every subset $Q \subseteq \mathbb{P}$ of original variables, the set $Q' = \{p' \ : \ p \in Q\} \subseteq \mathbb{P}'$ is its image, and the other way around. We suppose that $(.)'$ is an involution, i.e., it behaves as an identity when applied twice. Now, a **DL-PA** valuation extends to the form of $X \cup Y'$, where $X \subseteq \mathbb{P}$ and $Y' \subseteq \mathbb{P}'$. As a result, the definition of **DL-PA** validity expands to the power set of $\mathbb{P} \cup \mathbb{P}'$, i.e., $2^{\mathbb{P} \cup \mathbb{P}'}$.

Remark 3.1 *We embed the* **D-HT** *model (H, T) into* **DL-PA** *logic as a valuation $H \cup T'$: H encodes the here-valuation while T' encodes the there-valuation. On*

$$\begin{aligned}
\texttt{mkFalse}^{\geq 0}(\{p_1,\ldots,p_n\}) &= (p_1{:=}\bot \cup \texttt{skip});\ \cdots\ ;(p_n{:=}\bot \cup \texttt{skip})\\
\texttt{mkFalse}^{>0}(\{p_1,\ldots,p_n\}) &= (p_1{:=}\bot \cup \cdots \cup p_n{:=}\bot);\texttt{mkFalse}^{\geq 0}(P')\\
\texttt{cp}(\{\texttt{p}_1,\ldots,\texttt{p}_\texttt{n}\}) &= p'_1{:=}p_1;\ \cdots\ ;p'_n{:=}p_n\\
\texttt{cpBack}(\{\texttt{p}_1,\ldots,\texttt{p}_\texttt{n}\}) &= p_1{:=}p'_1;\ \cdots\ ;p_n{:=}p'_n
\end{aligned}$$

Table 3.3: Fundamental units (**DL-PA** programs) of embedding.

the other hand, we can translate a **DL-PA** *model* $X \cup Y'$ *into* **D-HT** *logic under some constraints: note that in order to respect the heredity constraint hidden in the structure of HT models, our translation has to guarantee that* X *is a subset of* Y*. Therefore, we can just translate* **DL-PA** *models,* $X \cup Y'$ *such that* $X \subseteq Y$*, and we get a* **D-HT** *model* (X, Y)*.*

3.3.2 Molecular DL-PA programs of embedding

Table 3.3 collects some **DL-PA** programs that are going to be useful for our enterprise. In that table, $P = \{p_1, \ldots, p_n\}$ is a finite (possibly empty) subset of $\mathbb{P}$ and each p'_i is a copy of p_i as explained above. Also note that $P' \subset P$ is some set obtained from P removing exactly one propositional variable. However, we can easily generalise the definitions given in Table 3.3 to ones where the domain P is a countable subset of $\mathbb{P}$. In both cases, when P is empty, we stipulate that all these programs except $\texttt{mkFalse}^{>0}(\emptyset)$ trivially equal `skip`. However, $\texttt{mkFalse}^{>0}(\emptyset)$ is inexecutable in this case.

Let $P = \{p_1, \ldots, p_n\}$. The program $\texttt{mkFalse}^{\geq 0}(P)$ nondeterministically makes some of the variables of P false, possibly none. The program $\texttt{mkFalse}^{>0}(P)$ nondeterministically makes false at least one of the variables of P, and possibly more. Its subprogram $p_1{:=}\bot \cup \cdots \cup p_n{:=}\bot$ makes exactly one of the variables in the valuation P false. The program $\texttt{cp}(\texttt{P})$ assigns to each 'fresh' variable p'_i the truth value of p_i, while the program $\texttt{cpBack}(\texttt{P})$ assigns to each variable p_i the truth value of p'_i. More explicitly, for each $i \in \{1, \ldots, n\}$, $\texttt{cp}(\texttt{P})$ produces the fresh variable, p'_i on the valuation V (where it is applied) if $p_i \in V$; otherwise removes p'_i, and the same procedure applies conversely for $\texttt{cpBack}(\texttt{P})$. For our purposes, we shall use the former as a way of storing the truth value of each variable of P before they undergo some changes. That will allow later on to retrieve the original values of the variables in P by means of the $\texttt{cpBack}(\texttt{P})$ program. Therefore the sequence $\texttt{cp}(\texttt{P});\texttt{cpBack}(\texttt{P})$ leaves the truth values of the variables in P unchanged.

Observe that each program of Table 3.3 has length linear in the cardinality

of P. Observe also that the programs $\texttt{mkFalse}^{\geq 0}(P)$ and $\texttt{mkFalse}^{>0}(P)$ are non-deterministic. Moreover, as $\texttt{mkFalse}^{\geq 0}(P)$ is always executable, $\texttt{mkFalse}^{>0}(P)$ is not because the latter cannot be applied to empty valuations. In contrast, the programs $\texttt{cp(P)}$ and $\texttt{cpBack(P)}$ are deterministic and always executable: $[\texttt{cp(P)}]\varphi$ and $\langle\texttt{cp(P)}\rangle\varphi$ are equivalent, as well as $[\texttt{cpBack(P)}]\varphi$ and $\langle\texttt{cpBack(P)}\rangle\varphi$.

Lemma 3.1 (Program Lemma) *Let $P \subseteq \mathbb{P}$ be non-empty. Then*

$$\begin{aligned}
\|\texttt{mkFalse}^{\geq 0}(P)\|_{\textbf{DL-PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') : X_2 = X_1 \setminus Q, \text{ for some } Q \subseteq P \text{ and } Y_2' = Y_1'\} \\
\|\texttt{mkFalse}^{>0}(P)\|_{\textbf{DL-PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') : X_2 = X_1 \setminus Q, \text{ for some } Q \subseteq P \text{ and } Y_2' = Y_1'\} \\
\|\texttt{cp(P)}\|_{\textbf{DL-PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') : X_2 = X_1 \text{ and } Y_2' = (X_1 \cap P)' \cup (Y_1' \setminus P')'\} \\
\|\texttt{cpBack(P)}\|_{\textbf{DL-PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') : X_2 = (Y_1' \cap P')' \cup (X_1 \setminus P) \text{ and } Y_2' = Y_1'\}.
\end{aligned}$$

It follows from the interpretations of $\texttt{cp(P)}$ and $\texttt{mkFalse}^{\geq 0}(P)$ that

$$\begin{aligned}
&\|\texttt{cp(P)}; \texttt{mkFalse}^{\geq 0}(P)\|_{\textbf{DL-PA}} = \\
&\Big\{(X_1 \cup Y_1', X_2 \cup Y_2') : X_2 = X_1 \setminus Q \text{ for some } Q \subseteq P \text{ and } Y_2' = (X_1 \cap P)' \cup (Y_1' \setminus P')\Big\}.
\end{aligned}$$

Remark 3.2 *One should note that when $\texttt{mkFalse}^{\geq 0}(P)$ and $\texttt{mkFalse}^{>0}(P)$ are applied to a valuation V in which no variables of P exists, i.e., $V \cap P = \emptyset$, these programs always make a loop and we obtain the same valuation again. The same may occur even when $V \cap P$ is nonempty when one just makes false the propositional variables of $P \setminus V$ which are already false in V. The interpretation of such programs clearly differs when $P \subseteq V$ because having $\texttt{mkFalse}^{>0}(P)$ applied to V while we reach a state $V' \subset V$, we obtain a valuation $V'' \subseteq V$ after $\texttt{mkFalse}^{\geq 0}(P)$ is applied to V. As a result, for nonempty $P \subseteq \mathbb{P}$,*

$$\texttt{mkFalse}^{>0}(P) = \texttt{mkFalse}^{\geq 0}(P) \setminus \{(V, V) \ : \ P \subseteq V \subseteq \mathbb{P}\}.$$

At this point, just before we pass to the next subsection, it is also useful to note that:

$$\|\texttt{skip}\|_{\textbf{DL-PA}} = \|\top?\|_{\textbf{DL-PA}} = \|p{:=}p\|_{\textbf{DL-PA}} = \{(V, V) \ : \ V \subseteq \mathbb{P}\}.$$

3.3.3 Translating $\mathcal{L}_{\textbf{D-HT}}$ to $\mathcal{L}_{\textbf{DL-PA}}$

To start with we translate the formulas and the programs of the language $\mathcal{L}_{\textbf{D-HT}}$ into the language $\mathcal{L}_{\textbf{DL-PA}}$. The translation is given in Table 3.4 in terms of a recursively defined mapping tr, where we have omitted the homomorphic cases such as $\mathsf{tr}([\pi]\varphi) = [\mathsf{tr}(\pi)]\mathsf{tr}(\varphi)$ and $\mathsf{tr}(\varphi?) = \Big(\mathsf{tr}(\varphi)\Big)?$. Observe that tr is polynomial.

$$
\begin{aligned}
\mathsf{tr}(p) &= p, \qquad \text{for } p \in \mathbb{P} \\
\mathsf{tr}(\bot) &= \bot \\
\mathsf{tr}(\varphi \to \psi) &= [\mathtt{skip} \cup \mathtt{cpBack}(\mathbb{P}_{\varphi\to\psi})]\Big(\mathsf{tr}(\varphi) \supset \mathsf{tr}(\psi)\Big) \\
\mathsf{tr}(+p) &= \Big(\sim p'? \ ; p' {:=} \top\Big) \cup \Big(\sim p \wedge p'? \ ; p {:=} \top\Big) \\
\mathsf{tr}(-p) &= \Big(p? \ ; p {:=} \bot\Big) \cup \Big(\sim p \wedge p'? \ ; p' {:=} \bot\Big)
\end{aligned}
$$

Table 3.4: Translation from $\mathcal{L}_{\text{D-HT}}$ into $\mathcal{L}_{\text{DL-PA}}$.

For example,

$$
\begin{aligned}
\mathsf{tr}(\top) = \mathsf{tr}(\bot \to \bot) &= [\mathtt{skip} \cup \mathtt{skip}]\top \\
\mathsf{tr}(p \vee \neg p) &= p \vee [\mathtt{skip} \cup p{:=}p']{\sim}p \\
\mathsf{tr}(p \to q) &= [\mathtt{skip} \cup (p{:=}p' \ ; q{:=}q')](p \supset q).
\end{aligned}
$$

The first formula is equivalent to $\top$, and the second is equivalent to $p \vee (\sim p \wedge \sim p')$, i.e., to $p \vee \sim p'$. Finally, the third is equivalent to $(p \supset q) \wedge (p' \supset q')$.

Lemma 3.2 (Main Lemma) $(H, T) \in \|\varphi\|_{\text{D-HT}}$ *iff* $H \cup T' \in \|\mathsf{tr}(\varphi)\|_{\text{DL-PA}}$.

Proof. See Lemma A.3 and its proof in Section A.3 of Appendix A. q.e.d.

3.3.4 From D-HT to DL-PA

We now establish how tr can be used to prove that a given formula φ is **D-HT** satisfiable. To that end, we prefix the translation by the '$\mathtt{cp}(\mathbb{P}_\varphi)$' program that is followed by the '$\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)$' program. The '$\mathtt{cp}(\mathbb{P}_\varphi)$' program produces a 'classical' valuation $T \cup (T \cap \mathbb{P}_\varphi)'$, for some subset T of $\mathbb{P}$ (as far as the variables of φ are concerned), and then '$\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)$' program transforms the valuation $T \cup (T \cap \mathbb{P}_\varphi)'$ into a valuation $H \cup (T \cap \mathbb{P}_\varphi)'$ for some H such that $H \subseteq T$.

Theorem 3.1 *Let φ be an $\mathcal{L}_{\text{D-HT}}$ formula. Then we have:*

- *φ is* **D-HT** *satisfiable if and only if* $\langle \mathtt{cp}(\mathbb{P}_\varphi)\rangle\langle \mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)\rangle \mathsf{tr}(\varphi)$ *is* **DL-PA** *satisfiable.*
- *φ is* **D-HT** *valid if and only if* $[\mathtt{cp}(\mathbb{P}_\varphi)][\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)]\mathsf{tr}(\varphi)$ *is* **DL-PA** *valid.*

The proof of this theorem is given by the Main Lemma and the Program Lemma.

As a result of Theorem 3.1, the formula $[\mathtt{cp}(\{\mathtt{p}\})][\mathtt{mkFalse}^{\geq 0}(\{p\})]\mathsf{tr}(p \vee \neg p)$ should not be **DL-PA** valid since we know that $p \vee \neg p$ is not **D-HT** valid. We indeed have the following sequence of equivalent formulas:

1. $[\mathtt{cp}(\{\mathtt{p}\})][\mathtt{mkFalse}^{\geq 0}(\{p\})]\mathsf{tr}(p \vee \neg p)$
2. $[p'{:=}p][p{:=}\bot \cup \mathtt{skip}](p \vee [\mathtt{skip} \cup p{:=}p']{\sim}p)$
3. $[p'{:=}p][p{:=}\bot \cup \mathtt{skip}](p \vee {\sim}p')$
4. $[p'{:=}p]\Big([p{:=}\bot](p \vee {\sim}p') \wedge (p \vee {\sim}p')\Big)$
5. $[p'{:=}p]\Big({\sim}p' \wedge (p \vee {\sim}p')\Big)$
6. ${\sim}p \wedge (p \vee {\sim}p)$
7. ${\sim}p$

The last is obviously not **DL-PA** valid, so the first line is not **DL-PA** valid either.

Theorem 3.2 *For an $\mathcal{L}_{\text{D-HT}}$ formula φ,*

$$\models_{\text{D-HT}} \varphi \text{ if and only if } \models_{\text{DL-PA}} [\mathtt{cp}(\mathbb{P}_\varphi); \mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)]\mathsf{tr}(\varphi).$$

3.3.5 From dynamic equilibrium logic to DL-PA

Having seen how **D-HT** can be embedded into **DL-PA**, we now turn to equilibrium logic.

Theorem 3.3 *For every $\mathcal{L}_{\text{D-HT}}$ formula χ, $T \subseteq \mathbb{P}$ is an equilibrium model of χ if and only if $T \cup T'$ is a* **DL-PA** *model of $\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}(\chi)$.*

Proof. See Theorem A.4 and its proof in Section A.3 of Appendix A. q.e.d.

Theorem 3.4 *Let χ and φ be $\mathcal{L}_{\text{D-HT}}$ formulas. Then $\chi \mathrel{|\!\approx} \varphi$ if and only if*

$$\langle \mathtt{cp}(\mathbb{P}_\chi \cup \mathbb{P}_\varphi)\rangle \Big(\big(\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}(\chi)\big) \supset \mathsf{tr}(\varphi) \Big)$$

is **DL-PA** *valid.*

Proof. See Theorem A.5 and its proof in Section A.3 of Appendix A. q.e.d.

Theorem 3.4 provides a polynomial embedding of the consequence problem in our dynamic equilibrium logic into **DL-PA**.

$$
\begin{aligned}
\mathsf{tr}'(p) &= p, \qquad \text{for } p \in \mathbb{P} \\
\mathsf{tr}'(\varphi \supset \psi) &= \mathsf{tr}'(\varphi) \to \mathsf{tr}'(\psi) \\
\mathsf{tr}'(p{:=}\top) &= p? \cup \big(+p \;;+p\big) \\
\mathsf{tr}'(p{:=}\bot) &= \neg p? \cup \big(-p \;;-p\big)
\end{aligned}
$$

Table 3.5: Translation from **DL-PA** with assignments only to $\top$ and $\bot$ into $\mathcal{L}_{\textbf{D-HT}}$ (main cases).

3.3.6 From DL-PA to D-HT

In this subsection we present a simple translation of the fragment of **DL-PA** whose atomic assignment programs are restricted to $p{:=}\top$ and $p{:=}\bot$ and with the conversion operator:

The translation is given in Table 3.5, where we have omitted the homomorphic cases. In the last two lines, $\mathsf{tr}'(p{:=}\top)$ makes p true both here and there, while $\mathsf{tr}'(p{:=}\bot)$ makes p false both here and there. The translation is clearly polynomial.

So, the nondeterministic program $+_\forall p \cup -_\forall p$ makes p behave classically: after its execution, the truth value of p is the same here and there. Note that both $+_\forall p$ and $-_\forall p$ are always executable.

The next lemma is the analog of the Main Lemma adapted to tr', and is used in the proof of Theorem 3.5.

Lemma 3.3 *Let φ be a* **DL-PA** *formula. Then,*

$$V \in \|\varphi\|_{\textbf{DL-PA}} \quad \textit{if and only if} \quad (V,V) \in \|\mathsf{tr}'(\varphi)\|_{\textbf{D-HT}}.$$

Now, we are ready to show how tr' can be used to prove that a given formula φ is **DL-PA** satisfiable.

Theorem 3.5 *Let φ be a* **DL-PA** *formula. Then, φ is* **DL-PA** *satisfiable if and only if $\mathsf{tr}'(\varphi) \wedge \bigwedge_{p \in \mathbb{P}_\varphi}(p \vee \neg p)$ is satisfiable in* **D-HT**.

3.4 Conclusion and future work

We have seen in this chapter to update an HT model by simple programs, but what about updates by complex programs? Actually we may implement such updates by means of complex **D-HT** programs. For example, the **D-HT** program

$$(\neg p \vee q)? \cup (-p; +q)$$

makes the implication $p \to q$ true, whatever the initial HT model is (although there may be other minimal ways of achieving this). More generally, let us consider that an abstract semantical update operation is a function $f : \mathbb{HT} \longrightarrow 2^{\mathbb{HT}}$ associating to every HT model (H, T) the set of HT models $f(H, T)$ resulting from the update. If the language is finite then for every such f we can design a program π_f such that $\|\pi_f\|_{\mathbf{D\text{-}HT}} = f$, viz. the graph of f. This makes use of the fact that in particular we can uniquely (up to logical equivalence) characterise HT models by means of the corresponding formulas. For example, the formula $\Big(\langle +p\rangle\top \wedge \langle -p\rangle\top\Big) \wedge \Big(\bigwedge_{q\neq p} \neg q\Big)$ identifies the HT model $(\emptyset, \{p\})$. Note finally that we cannot express the HT model $(\emptyset, \{p\})$ in the language $\mathcal{L}_{\mathbf{HT}}$, where there is no formula distinguishing that model from the model $(\{p\}, \{p\})$.

We plan as a future work to investigate the relation between **MEM** logic and **DL-PA**, and how these logics can be combined.

Chapter 4

From Epistemic Specifications to Epistemic ASP

As an alternative to the approach proposed in the previous chapter, this chapter presents an epistemic extension of equilibrium logic. We also compare our approach with Gelfond's epistemic specifications, mentioned in Section 1.4 of Chapter 1. Given that equilibrium logic is based on **HT** logic, the first step is to extend **HT** logic by two epistemic operators K and $\hat{\mathsf{K}}$ allowing to talk about knowledge and belief. The latter is slightly different from M. Equilibrium models and the equilibrium consequence (entailment) relation are then defined in the standard way. We respectively call the resulting formalisms *epistemic here-and-there logic* (abbreviated by **E-HT**), and *epistemic equilibrium logic*. Afterwards, we also define another epistemic extension of equilibrium logic called *autoepistemic equilibrium logic* which is based on the previous one. We finally give a strong equivalence result for EHT theories.

The rest of the chapter is organised as follows. Section 4.1 introduces epistemic here-and-there logic (**E-HT**): we give language, semantics and some semantical properties. Section 4.2 defines epistemic equilibrium models and epistemic equilibrium consequence, and also autoepistemic equilibrium models, and show that the notion of strong equivalence can be captured in **E-HT**: the strong equivalence of two EHT theories in the sense of epistemic equilibrium models can be verified by checking their equivalence in **E-HT** logic. Finally, Section 4.5 concludes with some final remarks and discussions about future work.

4.1 An epistemic extension of HT logic

In this section we introduce epistemic **HT** logic, abbreviated **E-HT**. Its language extends that of **HT** logic by two epistemic modal operators K and $\hat{\mathsf{K}}$. Its models

generalise HT models (H, T) to collections $\{(H_i, T_i)\}_i$ of such models. An S5 model is a nonempty set of valuations, so an EHT model can also be viewed as a refinement of S5 models where possible worlds are replaced by HT models.

4.1.1 Language ($\mathcal{L}_{\textbf{E-HT}}$)

The language $\mathcal{L}_{\textbf{E-HT}}$ is given by the following grammar:

$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \mathsf{K}\varphi \mid \hat{\mathsf{K}}\varphi$$

where p ranges over a countably infinite set $\mathbb{P} = \{p, q, \ldots\}$ of propositional variables. A formula is said to be *objective* (or *non-modal*) if it does not contain modal operators K or $\hat{\mathsf{K}}$. As usual, $\top$, $\neg\varphi$ and $\varphi \leftrightarrow \psi$ respectively abbreviate $\bot \to \bot$, $\varphi \to \bot$ and $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Given a formula φ, the set of variables that appear in φ is noted $\mathbb{P}_\varphi$. For example, $\mathbb{P}_{\mathsf{K}(p \to \neg q)} = \{p, q\}$. An EHT *theory* Φ is a finite set of formulas in the language $\mathcal{L}_{\textbf{E-HT}}$ of **E-HT** logic. The set of variables occurring in Φ is $\mathbb{P}_\Phi = \bigcup_{\varphi \in \Phi} \mathbb{P}_\varphi$.

4.1.2 Epistemic here-and-there models

An *epistemic here-and-there (EHT) model* is an ordered pair whose first component is a nonempty collection of *there-valuations*, and the second component assigns a *here-valuation* to each there-valuation in the first component. Formally, an EHT model is an ordered pair $(\mathcal{T}, \hbar)$ in which

- $\mathcal{T} \subseteq 2^{\mathbb{P}}$ is a nonempty set of valuations;
- $\hbar$ is a function $\hbar : \mathcal{T} \to 2^{\mathbb{P}}$ such that $\hbar(T) \subseteq T$ for every $T \in \mathcal{T}$.

An EHT model can be viewed as a collection of HT models: the *here-function* $\hbar$ associates to each valuation $T \in \mathcal{T}$ the HT model $(\hbar(T), T)$. In particular, a non-epistemic HT model can be identified with an EHT model where $\mathcal{T}$ is a singleton. The inclusion constraint on the function $\hbar$ is the *heredity constraint* of intuitionistic logic. A particular case is when $\hbar$ is the identity function, i.e., $\hbar = id$. Then $\hbar(T) = T$ for every $T \in \mathcal{T}$, which means that such EHT models are nothing but **S5** models. We write $\mathbb{E}$-$\mathbb{HT}$ for the set of all EHT models, with typical elements $\mathfrak{M}$, $\mathfrak{M}'$, etc.

A pointed EHT model is a pair $\big((\mathcal{T}, \hbar), T\big)$ such that $(\mathcal{T}, \hbar)$ is an EHT model and $T \in \mathcal{T}$ (the *actual world*). It will sometimes be useful to have the alternative notation $\{(\hbar(T), T)\}_{T \in \mathcal{T}}$ for EHT models and $(\{(\hbar(T), T)\}_{T \in \mathcal{T}}, (\hbar(T), T))$ for pointed EHT models.

4.1.3 Truth conditions

Now, we define the EHT satisfaction relation.[1]

$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} p$ if $p \in \hbar(T)$;
$(\mathcal{T},\hbar),T \not\models_{\textbf{E-HT}} \bot$;
$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi \wedge \psi$ if $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi$ and $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \psi$;
$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi \vee \psi$ if $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi$ or $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \psi$;
$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi \rightarrow \psi$ if $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi \supset \psi$ and $(\mathcal{T},id),T \models_{\textbf{E-HT}} \varphi \supset \psi$;
$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \mathsf{K}\varphi$ if $(\mathcal{T},\hbar),T' \models_{\textbf{E-HT}} \varphi$ for every $T' \in \mathcal{T}$;
$(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \hat{\mathsf{K}}\varphi$ if $(\mathcal{T},\hbar),T' \models_{\textbf{E-HT}} \varphi$ for some $T' \in \mathcal{T}$.

It follows that $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \neg\varphi$ iff $(\mathcal{T},\hbar),T \not\models_{\textbf{E-HT}} \varphi$ and $(\mathcal{T},id),T \not\models_{\textbf{E-HT}} \varphi$. Due to the heredity property of intuitionistic logic this will be equivalent to $(\mathcal{T},id),T \not\models_{\textbf{E-HT}} \varphi$ (to be shown later).

When a formula is satisfied in all there-valuations we write $(\mathcal{T},\hbar) \models_{\textbf{E-HT}} \varphi$.

When the collection of possible HT models is represented explicitly then we underline the actual world. When no HT model is underlined then the relation holds for all HT models in the collection. Here are some examples.

1. $\Big\{(\emptyset,\{p\}),\underline{(\emptyset,\{q\})}\Big\} \models_{\textbf{E-HT}} \neg p$ since $\Big\{(\emptyset,\{p\}),\underline{(\emptyset,\{q\})}\Big\} \not\models_{\textbf{E-HT}} p$ and $\Big\{(\{p\},\{p\}),\underline{(\{q\},\{q\})}\Big\} \not\models_{\textbf{E-HT}} p$.
2. $\Big\{\underline{(\emptyset,\{p\})},(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \neg\neg p$.
3. $\Big\{(\emptyset,\{p\}),(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \mathsf{K}\neg r$ since $\Big\{(\emptyset,\{p\}),(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \neg r$.
4. $\Big\{(\emptyset,\{p\}),(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \neg\mathsf{K}\neg p$ since $\Big\{\underline{(\{p\},\{p\})},(\{q\},\{q\})\Big\} \not\models_{\textbf{E-HT}} \neg p$.
5. $\Big\{(\emptyset,\{p\}),(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \neg\neg\mathsf{K}(p \vee q)$ as $\Big\{(\{p\},\{p\}),(\{q\},\{q\})\Big\} \models_{\textbf{E-HT}} p \vee q$.
6. $\Big\{(\emptyset,\{p\}),(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \hat{\mathsf{K}}\neg\neg p$ since $\Big\{\underline{(\emptyset,\{p\})},(\emptyset,\{q\})\Big\} \models_{\textbf{E-HT}} \neg\neg p$.

An *EHT model* for φ is a pointed model $\Big((\mathcal{T},\hbar),T\Big)$ such that $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi$. When $\Big((\mathcal{T},\hbar),T\Big)$ is a model for φ for every $T \in \mathcal{T}$ then we say that it is *valid* in $(\mathcal{T},\hbar)$. If formulas of the form $\mathsf{K}\varphi$, $\neg\mathsf{K}\varphi$, $\neg\neg\mathsf{K}\varphi$, $\hat{\mathsf{K}}\varphi$, $\neg\hat{\mathsf{K}}\varphi$, and $\neg\neg\hat{\mathsf{K}}\varphi$ has a pointed model then it is also valid in that model. The set of all EHT models of an $\mathcal{L}_{\textbf{E-HT}}$ formula φ is noted $\|\varphi\|_{\textbf{E-HT}}$. This extends to sets of formulas. For example, $\|p \vee \neg p\|_{\textbf{E-HT}}$ is the collection of all EHT models $\Big((\mathcal{T},\hbar),T\Big)$ in which

[1] The material implication $\supset$ in the fifth clause is just a shorthand enabling a concise formulation; its truth condition is: $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \varphi \supset \psi$ iff $(\mathcal{T},\hbar),T \not\models_{\textbf{E-HT}} \varphi$ or $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \psi$.

$p \in \hbar(T)$ or $p \notin T$. Raising the bar of our examples, $\|\mathsf{K}\,(p \vee \neg p) \to \hat{\mathsf{K}}\,\neg\neg p\|_{\text{E-HT}}$ is the collection of all EHT models $(\mathcal{T}, \hbar)$ such that $p \in T$ for some $T \in \mathcal{T}$. Finally, $\|\neg\mathsf{K}\,\neg\neg\varphi \to \neg\mathsf{K}\,\varphi\|_{\text{E-HT}}$ equals $\mathbb{E}$-$\mathbb{HT}$.

Proposition 4.1 *Given an $\mathcal{L}_{\text{E-HT}}$ formula φ and an E-HT model $(\mathcal{T}, \hbar)$, let $P_T \subseteq \mathbb{P}$ be such that $P_T \cap \mathbb{P}_\varphi = \emptyset$ for every $T \in \mathcal{T}$. Then, for every $Q_T \subseteq P_T$ $(T \in \mathcal{T})$,*

$$\Big((\mathcal{T}, \hbar), T_0\Big) \in \|\varphi\|_{\text{E-HT}} \quad \textit{if and only if} \quad \Big((\mathcal{T}', \hbar'), T_0'\Big) \in \|\varphi\|_{\text{E-HT}}$$

where $\mathcal{T}' = \{T \cup P_T \ : \ T \in \mathcal{T}\}$ and $\hbar'(T') = \hbar(T) \cup Q_T$ for every $T' \in \mathcal{T}'$ and $T \in \mathcal{T}$.

The proof is by induction on φ.

4.1.4 EHT validity

A formula φ is EHT *satisfiable* if $\|\varphi\|_{\text{E-HT}} \neq \emptyset$. It is EHT *valid* if $\|\varphi\|_{\text{D-HT}} = \mathbb{E}$-$\mathbb{HT}$.

All the principles of the intuitionistic modal logics that were studied in the literature (Bierman and de Paiva [2000]; Fariñas del Cerro and Raggio [1983]; Fischer-Servi [1976]; Simpson [1994]) are EHT valid. The axiom schemas for intuitionistic **S5** of Simpson [1994] are listed in Table 4.1. The inference rules are modus ponens and necessitation.

0.	All substitution instances of theorems of intuitionistic propositional logic
1.	$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
2.	$\Box(\varphi \to \psi) \to (\Diamond\varphi \to \Diamond\psi)$
3.	$\neg\Diamond\bot$
4.	$\Diamond(\varphi \vee \psi) \to (\Diamond\varphi \vee \Diamond\psi)$
5.	$(\Diamond\varphi \to \Box\psi) \to \Box(\varphi \to \psi)$
6.	$\Diamond\top$
7.	$(\Box\varphi \to \varphi) \wedge (\varphi \to \Diamond\varphi)$
8.	$(\Diamond\Box\varphi \to \varphi) \wedge (\varphi \to \Box\Diamond\varphi)$
9.	$(\Box\varphi \to \Box\Box\varphi) \wedge (\Diamond\Diamond\varphi \to \Diamond\varphi)$
10.	$(\Diamond\Box\varphi \to \Box\varphi) \wedge (\Diamond\varphi \to \Box\Diamond\varphi)$

Table 4.1: Axiom schemas of intuitionistic **S5** (Simpson [1994]).

As always in intuitionistic modal logics, K and $\hat{\mathsf{K}}$ are not dual: while $\hat{\mathsf{K}}\,\varphi \to \neg\mathsf{K}\,\neg\varphi$ is valid, the other direction is not. Here are some other invalid schemas.

First, both $\neg\neg \mathsf{K}\,\varphi \to \mathsf{K}\,\varphi$ and $\mathsf{K}\,\neg\neg\varphi \to \mathsf{K}\,\varphi$ are **E-HT** invalid. The same holds if we replace K by $\hat{\mathsf{K}}$. (The EHT model $(\mathcal{T},\hbar)$ with $\mathcal{T} = \Big\{\{p\}\Big\}$ and $\hbar(\{p\}) = \emptyset$ provides a countermodel for all of them.) Second, $\neg\mathsf{K}\,\neg\varphi \to \neg\neg\mathsf{K}\,\varphi$ is invalid, too. The opposite direction is valid.

The following lemma says that if a formula has an EHT model then it also has a total EHT model (in other words, an S5 model). This is the *heredity* (*monotonicity*) *property* of intuitionistic logic.

Proposition 4.2 *Let* $\Big((\mathcal{T},\hbar),T\Big)$ *be an EHT model of* φ. *Then* $\Big((\mathcal{T},id),T\Big)$ *is also an EHT model of* φ.

We now list some useful equivalences.

Proposition 4.3 *Let* $\Big((\mathcal{T},\hbar),T\Big)$ *be an EHT model. Let* φ *be an* $\mathcal{L}_{\textbf{E-HT}}$ *formula.*

1. $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \neg\varphi$ *iff* $(\mathcal{T},id),T \not\models_{\textbf{E-HT}} \varphi$;
2. $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \neg\neg\varphi$ *iff* $(\mathcal{T},id),T \models_{\textbf{E-HT}} \varphi$;
3. $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \neg\mathsf{K}\,\varphi$ *iff* $(\mathcal{T},id),T' \not\models_{\textbf{E-HT}} \varphi$, *for some* $T' \in \mathcal{T}$;
4. $(\mathcal{T},\hbar),T \models_{\textbf{E-HT}} \neg\hat{\mathsf{K}}\,\varphi$ *iff* $(\mathcal{T},id),T' \not\models_{\textbf{E-HT}} \varphi$, *for any* $T' \in \mathcal{T}$.

Proof. See Proposition A.10 and its proof in Section A.4 of Appendix A. q.e.d.

Now, we are ready to make a short list of theorems of our logic **E-HT** that will come in handy for further discussions.

Proposition 4.4 *For each line below, the three formulas are EHT equivalent.*

$\hat{\mathsf{K}}\,\neg\neg\varphi$	$\neg\mathsf{K}\,\neg\varphi$	$\neg\neg\hat{\mathsf{K}}\,\varphi$
$\mathsf{K}\,\neg\neg\varphi$	$\neg\hat{\mathsf{K}}\,\neg\varphi$	$\neg\neg\mathsf{K}\,\varphi$
$\neg\hat{\mathsf{K}}\,\varphi$	$\mathsf{K}\,\neg\varphi$	$\neg\hat{\mathsf{K}}\,\neg\neg\varphi$
$\neg\mathsf{K}\,\varphi$	$\hat{\mathsf{K}}\,\neg\varphi$	$\neg\mathsf{K}\,\neg\neg\varphi$

Corollary 4.1 *The following are EHT valid while their converses are not.*

1. $\mathsf{K}\,\varphi \to \neg\hat{\mathsf{K}}\,\neg\varphi$
2. $\hat{\mathsf{K}}\,\varphi \to \neg\mathsf{K}\,\neg\varphi$
3. $\neg\hat{\mathsf{K}}\,\neg\varphi \to \neg\neg\hat{\mathsf{K}}\,\varphi$
4. $\neg\neg\mathsf{K}\,\varphi \to \neg\mathsf{K}\,\neg\varphi$

4.2 Epistemic equilibrium logic

An equilibrium model (EM) of a formula is a classical model satisfying a minimality condition when viewed as a total HT model. We here generalise such models from classical to S5 models and define epistemic equilibrium models (EEMs). The corresponding non-monotonic consequence relation is a conservative extension of the standard equilibrium consequence relation.

4.2.1 Total models and their weakening

An EHT model $(\mathcal{T}, \hbar)$ is called *total* if $\hbar = id$, where id refers to the identity function. A total EHT model corresponds to a classical **S5** model, so validity in classical **S5** is the same as validity in total EHT models. We may therefore identify $(\mathcal{T}, id), T \models_{\textbf{E-HT}} \varphi$ with $\mathcal{T}, T \models_{\textbf{S5}} \varphi$.

Given two EHT models $(\mathcal{T}_1, \hbar_1)$ and $(\mathcal{T}_2, \hbar_2)$, we write

$$(\mathcal{T}_1, \hbar_1) \trianglelefteq (\mathcal{T}_2, \hbar_2)$$

if $\mathcal{T}_1 = \mathcal{T}_2$ and $\hbar_1(T) \subseteq \hbar_2(T)$, for every $T \in \mathcal{T}$. We say that $(\mathcal{T}_1, \hbar_1)$ is weaker than $(\mathcal{T}_2, \hbar_2)$. This is a non-strict partial order. The corresponding strict partial order is defined in the standard way:

$$(\mathcal{T}_1, \hbar_1) \triangleleft (\mathcal{T}_2, \hbar_2) \text{ if } (\mathcal{T}_1, \hbar_1) \trianglelefteq (\mathcal{T}_2, \hbar_2) \text{ and } (\mathcal{T}_2, \hbar_2) \not\trianglelefteq (\mathcal{T}_1, \hbar_1).$$

4.2.2 Epistemic equilibrium models

An *epistemic equilibrium model* (EEM) of $\varphi \in \mathcal{L}_{\textbf{E-HT}}$ is a pointed set $\Big(\mathcal{T}, T\Big)$ where $\mathcal{T} \subseteq 2^{\mathbb{P}}$ is a nonempty set of valuations (i.e., a classical **S5** model) and $T \in \mathcal{T}$ is a distinguished element such that:

- $\mathcal{T}, T \models_{\textbf{S5}} \varphi$, and
- for any function $\hbar \ : \ \mathcal{T} \to 2^{\mathbb{P}}$ with $(\mathcal{T}, \hbar) \triangleleft (\mathcal{T}, id)$, $(\mathcal{T}, \hbar), T \not\models_{\textbf{E-HT}} \varphi$.

The first condition requires the total **E-HT** model $\Big((\mathcal{T}, id), T\Big)$ to be a model of φ. The second *minimality condition* requires that there is no E-HT model $\Big((\mathcal{T}, \hbar), T\Big)$ of φ that is strictly weaker than $\Big((\mathcal{T}, id), T\Big)$. As before, when $(\mathcal{T}, T) \models_{\textbf{S5}} \varphi$ for every $T \in \mathcal{T}$ then we just write $\mathcal{T} \models_{\textbf{S5}} \varphi$ and say that $\mathcal{T}$ is an epistemic equilibrium model of φ. These definitions are generalised to EHT theories in the standard way.

Proposition 4.5 *The following properties hold for* $\varphi \in \mathcal{L}_{\textbf{E-HT}}$.

1. *All EHT valid formulas have exactly one EEM, namely* $\{\emptyset\}$.

2. $\neg\varphi$ *has either a unique EEM* $\{\emptyset\}$ *or none: if* $\{\emptyset\} \models_{\mathbf{S5}} \varphi$ *then* $\{\emptyset\}$ *is the unique EEM of* $\neg\neg\varphi$ *and* $\neg\varphi$ *has none, otherwise vice versa.*

3. *If* φ *has no EEM then neither do* $\hat{\mathsf{K}}\varphi$ *and* $\mathsf{K}\varphi$.

4. *Let* $T \subseteq \mathbb{P}$ *be a valuation. For an objective (nonmodal)* φ *(i.e.,* $\varphi \in \mathcal{L}_{\mathbf{HT}}$*),* $\{T\}$ *and* $\{\emptyset, \underline{T}\}$ *are EEMs of* φ *if and only if* T *is an EM of* φ.

5. *For an objective* φ*, any arbitrary collection of EMs of* φ *is an EEM of* $\mathsf{K}\varphi$.

6. *For a nonmodal* $\mathcal{L}_{\mathbf{E\text{-}HT}}$ *formula* φ*,* T *is an EM of* φ *iff:*

   (a) $\{T\}$ *is an EEM of* $\hat{\mathsf{K}}\varphi$ *when* $\emptyset \models \varphi$.

   (b) $\{T\}$ *and* $\{\emptyset, T\}$ *are the EEMs of* $\hat{\mathsf{K}}\varphi$ *when* $\emptyset \not\models \varphi$.

Proof. See Proposition A.11 and its proof in Section A.4 of Appendix A. q.e.d.

The atomic p has a unique EM, namely $\{p\}$, so by the item 4 it has exactly two EEMs: $\{\{p\}\}$ and $\big\{\emptyset, \underline{\{p\}}\big\}$. Note that $p \vee \neg p$ has two EMs: $\emptyset$ and $\{p\}$, which means that $\emptyset \models p \vee \neg p$. So, $\{\emptyset\}$ and $\{\{p\}\}$ are the only EEMs of $\hat{\mathsf{K}}(p \vee \neg p)$, whereas $p \vee \neg p$ and $\mathsf{K}(p \vee \neg p)$ has one more: $\{\emptyset, \{p\}\}$ (see items 4 and 6a above). Remember that $\neg\neg p$ has no EMs, and as expected has no EEMs either because $\{\emptyset\} \not\models_{\mathbf{S5}} p$. Then, $\neg\neg p$, $\hat{\mathsf{K}}\neg\neg p$ and $\mathsf{K}\neg\neg p$ have no EEMs (see items 2 and 3 above), neither do $\neg\neg\hat{\mathsf{K}}p$, $\neg\mathsf{K}\neg p$, $\neg\neg\mathsf{K}p$ and $\neg\hat{\mathsf{K}}\neg p$ (see Proposition 4.4). According to the item 5, the EEMs of $\mathsf{K}(p \vee q)$ are: $\big\{\{p\}\big\}$, $\big\{\{q\}\big\}$, and $\big\{\{p\}, \{q\}\big\}$. Indeed, $p \vee q$ has just two EMs, namely $\{p\}$ and $\{q\}$. According to item 6b, the EEMs of $\hat{\mathsf{K}}(p \vee q)$ are: $\big\{\{p\}\big\}$, $\big\{\{q\}\big\}$, $\big\{\emptyset, \{p\}\big\}$, and $\big\{\emptyset, \{q\}\big\}$. Finally, while the EEMs of $\varphi \in \mathcal{L}_{\mathbf{HT}}$ and $\hat{\mathsf{K}}\varphi$ in general differ (see item 4 and item 6a), those of $\neg p$, $\hat{\mathsf{K}}\neg p$ and even $\mathsf{K}\neg p$ coincide since it is just the empty set. To illustrate item 2, $\neg\mathsf{K}p$ and $\neg\mathsf{K}\neg\neg p$ both have exactly one EEM, viz. $\{\emptyset\}$.

4.2.2.1 Consequence relation of epistemic equilibrium logic

For two $\mathcal{L}_{\mathbf{E\text{-}HT}}$ formulas ψ and φ, we say that φ is a *consequence* of ψ in epistemic equilibrium models, written $\psi \mid\!\approx \varphi$, if and only if for every epistemic equilibrium model $\big(\mathcal{T}, T_0\big)$ of ψ, $\big((\mathcal{T}, id), T_0\big)$ is an E-HT model of φ.

Proposition 4.6 *The following properties hold: for* $\varphi, \psi \in \mathcal{L}_{\mathbf{HT}}$*,*

1. *if* $\varphi \mid\!\approx \psi$ *then* $\mathsf{K}\varphi \mid\!\approx \mathsf{K}\psi$ *and* $\mathsf{K}\varphi \mid\!\approx \psi$.

2. *if* $\varphi \mid\!\approx \psi$ *then* $\hat{\mathsf{K}}\varphi \mid\!\approx \hat{\mathsf{K}}\psi$ *and* $\varphi \mid\!\approx \hat{\mathsf{K}}\psi$.

PROOF. See Proposition A.12 and its proof in Section A.4 of Appendix A. q.e.d.

Here are some examples:

- $\mathsf{K}(\neg p \to q) \mid\!\approx \mathsf{K}q$ and $\mathsf{K}(\neg p \to q) \mid\!\approx q$ because $\{q\}$ is the only equilibrium model of $\mathsf{K}(\neg p \to q)$ and $(\{q\}, \{q\})$ satisfies both $\mathsf{K}q$ and q: indeed we have seen before that $\neg p \to q \mid\!\approx q$, and hence by Lemma 4.6.1 we get the same result.

- $\neg p \to q \not\mid\!\approx \mathsf{K}q$ because $\Big\{\emptyset, \underline{\{q\}}\Big\}$ is an epistemic equilibrium model of $\neg p \to q$, yet $\Big\{(\emptyset, \emptyset), \underline{\{q\}, \{q\}}\Big\} \not\models_{\textbf{E-HT}} \mathsf{K}q$.

- We have $\hat{\mathsf{K}}p \mid\!\approx \hat{\mathsf{K}}\neg q$, $\hat{\mathsf{K}}p \mid\!\approx \neg q$ and $p \mid\!\approx \hat{\mathsf{K}}\neg q$, and a similar result also applies for K operator: the first two relations hold because $\mathcal{T}_1 = \Big\{\{p\}\Big\}$ and $\mathcal{T}_2 = \Big\{\emptyset, \{p\}\Big\}$ are the only epistemic equilibrium models of $\hat{\mathsf{K}}p$. Moreover, $(\mathcal{T}_1, id)$ and $(\mathcal{T}_2, id)$ both satisfy $\hat{\mathsf{K}}\neg q$ and $\neg q$. As for the last relation, the epistemic equilibrium models of p are: $\{p\}$ and $\Big\{\emptyset, \underline{\{p\}}\Big\}$. Moreover, $\mathcal{T}_1 \models_{\textbf{S5}} \hat{\mathsf{K}}\neg q$ and $\mathcal{T}_2, \{p\} \models_{\textbf{S5}} \hat{\mathsf{K}}\neg q$ both hold. On the other hand, we have seen above that $p \mid\!\approx \neg q$ holds. Hence, Lemma 4.6.2 confirms the first and the last results. The verification of K operator is similar.

- While $\hat{\mathsf{K}}(p \wedge q) \mid\!\approx \hat{\mathsf{K}}(p \vee q)$ and $p \wedge q \mid\!\approx \hat{\mathsf{K}}(p \vee q)$, we have $\hat{\mathsf{K}}(p \wedge q) \not\mid\!\approx p \vee q$: the latter doesn't hold because $\Big\{\{p, q\}, \underline{\emptyset}\Big\}$ is an epistemic equilibrium model of $\hat{\mathsf{K}}(p \wedge q)$, yet $\Big\{(\{p, q\}, \{p, q\}), \underline{(\emptyset, \emptyset)}\Big\}$ is not an E-HT model of $p \vee q$.

Given $\varphi \mid\!\approx \psi$ for $\varphi, \psi \in \mathcal{L}_{\textbf{HT}}$, we always have $\mathsf{K}\varphi \mid\!\approx \hat{\mathsf{K}}\psi$, but not necessarily the converse. The proof of the first claim is similar to the previous proofs, so we leave it to the reader. Now, let us see some counterexamples for the latter.

- We have seen before that $\neg q \mid\!\approx p \to q$. Moreover, we also have $\mathsf{K}\neg q \mid\!\approx \hat{\mathsf{K}}(p \to q)$ and $\hat{\mathsf{K}}\neg q \mid\!\approx \mathsf{K}(p \to q)$.

- However, while $p \mid\!\approx p$ and $\mathsf{K}p \mid\!\approx \hat{\mathsf{K}}p$, $\hat{\mathsf{K}}p \not\mid\!\approx \mathsf{K}p$ since $\Big\{\emptyset, \{p\}\Big\} \not\models_{\textbf{S5}} \mathsf{K}p$.

- Moreover, $\mathsf{K}(p \vee q) \mid\!\approx \hat{\mathsf{K}}\Big((p \vee q) \wedge \neg(p \wedge q)\Big)$, but not when K and $\hat{\mathsf{K}}$ are reversed.

Finally, we give two more examples: $\top \mid\!\approx \mathsf{K}\neg p$ which can be interpreted "in the absence of information, it is known that there is no evidence about p". This consequence is equivalent to $\top \mid\!\approx \neg\hat{\mathsf{K}}p$ and it also confirms the former interpretation

since it says "in the absence of information, p cannot be believed". One other example is: $\neg p \mid\!\approx \neg \mathsf{K}\, p$ which can be further interpreted as: "in lack of evidence for p, p cannot be known".

4.2.3 Strong equivalence

We now show that the equivalence of two EHT theories Φ and Ψ captures that the extensions of Φ and Ψ by an EHT theory Θ have the same EEMs, in other words, Φ and Ψ are strongly equivalent in the sense of epistemic equilibrium models. Our proof is a non-trivial generalisation of that of Lifschitz et al. [2001] from HT models to EHT models, which requires the characterisation of HT models.

Using Proposition 4.1, we may suppose without loss of generality that $\mathbb{P}$ is *finite*: this will allow us to describe HT models by formulas.

For an EHT model $(\mathcal{T}, \hbar)$ and $T \in \mathcal{T}$, we first define

$$\chi_T = \left(\bigwedge_{p \in T} p\right) \wedge \left(\bigwedge_{p \notin T} \neg p\right) \quad \text{and}$$

$$\eta_{T,\hbar} = \left(\bigwedge_{p \in \hbar(T)} p\right) \wedge \left(\bigwedge_{p \notin T} \neg p\right) \wedge \left(\bigwedge_{p \in T \setminus \hbar(T)} \neg\neg p\right) \wedge \left(\left(\bigvee_{p \in T \setminus \hbar(T)} p\right) \rightarrow \bigwedge_{T \in \mathcal{T}} \hat{\mathsf{K}}\, \chi_T\right).$$

Then we have: when χ_T is true at a pointed model $\big((\mathcal{T}, \hbar'), T'\big)$ then $\hbar'(T') = T' = T$. When $\eta_{T,\hbar}$ is true at $\big((\mathcal{T}, \hbar'), T'\big)$ due to the first two conjuncts, $\hbar(T) \subseteq \hbar'(T') \subseteq T' \subseteq T$ and then the third conjunct gives $T' = T$. Finally, the last conjunct adds that if $\hbar'(T') \neq \hbar(T)$ then $\hbar' = id$, otherwise $\hbar'(T') = \hbar(T)$, which futher implies $(\hbar'(T), T') = (\hbar(T), T)$.

Lemma 4.1 *Let $(\mathcal{T}, \hbar)$ be an EHT model.*

1. *If $(\mathcal{T}, \hbar'), T' \models_{\textbf{E-HT}} \{\hat{\mathsf{K}}\, \chi_T \ : \ T \in \mathcal{T}\}$ then $\hbar' = id$.*
2. *If $(\mathcal{T}, \hbar'), T' \models_{\textbf{E-HT}} \{\hat{\mathsf{K}}\, \eta_{T,\hbar} \ : \ T \in \mathcal{T}\}$ then $\hbar' = id$ or $\hbar' = \hbar$.*

We now give the main result.

Theorem 4.1 *The following conditions are equivalent:*

1. $\|\Phi\|_{\textbf{E-HT}} = \|\Psi\|_{\textbf{E-HT}}$.
2. *$\Phi \cup \Theta$ and $\Psi \cup \Theta$ have the same EEMs, for every Θ.*

Proof. See Theorem A.6 and its proof in Section A.4 of Appendix A. q.e.d.

4.3 Autoepistemic equilibrium logic

Based on epistemic equilibrium models of a formula, we first define autoepistemic equilibrium models by maximising the set of possible worlds and then prove a strong equivalence result for them.

4.3.1 Autoepistemic equilibrium models

Let $(\mathcal{T}, T)$ be an epistemic equilibrium model of φ. $(\mathcal{T}, T)$ is an *autoepistemic equilibrium model* of φ if there is no $\mathcal{T}'$ with $\mathcal{T} \subset \mathcal{T}'$ such that $(\mathcal{T}', T)$ is an epistemic equilibrium model of φ. We call the second condition the *maximality condition*: there is no bigger epistemic equilibrium model of φ. $\Big(\mathcal{T}, T\Big)$ is said to be an autoepistemic equilibrium model of an EHT theory Φ if it is an autoepistemic equilibrium model of all formulas in Φ.

The AEEM notion enables us to capture the full information obtained from an EHT theory, whereas the EEM concept is not strong enough to reveal the differences between EHT theories. Intuitively, the EHT formulas $\mathsf{K}\,p$, $\hat{\mathsf{K}}\,p$ and p respectively mean that p is true everywhere, somewhere and in the specified actual world of the collection. The AEEMs of $\mathsf{K}\,p$, $\hat{\mathsf{K}}\,p$ and p are respectively $\{\{p\}\}$, $\{\emptyset, \{p\}\}$ and $\{\emptyset, \underline{\{p\}}\}$. Note that $\{\{p\}\}$ is an EEM of all three, but is not able to distinguish p and $\hat{\mathsf{K}}\,p$ and $\mathsf{K}\,p$. Therefore, although they are minimal, they do not include much information. One reason for this is because it is a singleton model. However, $\{\emptyset, \underline{\{p\}}\}$ explicitly shows that we work with S5 models, and in this collection, p is true indeed in the actual world. Similarly, $\{\emptyset, \{p\}\}$ is strong enough to specify that $\mathsf{M}\,p$ describes a collection of worlds in which p is true in at least one. Here is another example. The EHT theory $\Phi = \{p, \mathsf{K}\,p \rightarrow (q \vee r)\}$ has four EEMs: $\Big\{\{p, q\}\Big\}$, $\Big\{\{p, r\}\Big\}$, $\Big\{\{p, q\}, \{p, r\}\Big\}$ and $\Big\{\emptyset, \underline{\{p\}}\Big\}$, but we choose last two as its AEEMs.

Proposition 4.7 *Let φ be an objective $\mathcal{L}_{\mathbf{HT}}$ formula. Then there is at most one autoepistemic equilibrium model of $\mathsf{K}\,\varphi$, viz. the collection of all equilibrium models of φ.*

Here are some more examples.

- The unique autoepistemic equilibrium model of $\mathsf{K}\,(p \vee q)$ is $\Big\{\{p\}, \{q\}\Big\}$.
- The autoepistemic equilibrium models of $\hat{\mathsf{K}}\,(p \vee q)$ are $\Big\{\emptyset, \{p\}\Big\}$ and $\Big\{\emptyset, \{q\}\Big\}$.
- The formula $\mathsf{K}\,p \vee \mathsf{K}\,q$ has the same epistemic and autoepistemic equilibrium models: $\Big\{\{p\}\Big\}$ and $\Big\{\{q\}\Big\}$.

4.3.2 Strong equivalence

Beyond the characterisation of here-there pairs as in the proof for epistemic equilibrium models it requires the characterisation of all there-worlds by means of Jankov-Fine like formulas (Blackburn et al. [2001a]).

Theorem 4.2 *Let* Φ *and* Ψ *be two* **E-HT** *theories. The following are equivalent:*

1. $\|\Phi\|_{\text{E-HT}} = \|\Psi\|_{\text{E-HT}}$;
2. $\Phi \cup \Theta$ *and* $\Psi \cup \Theta$ *have the same autoepistemic equilibrium models, for all* Θ.

Proof. See Proposition A.7 and its proof in Section A.4 of Appendix A. q.e.d.

4.4 Related work

At this point, just before we conclude the section, we would like to wander around literature to see some other works that tries to embed Gelfond's epistemic specifications (Chen [1997]; Truszczyński [2011]; Wang and Zhang [2005]. Through this aim, we discuss the relations between AEEMs, Gelfond's world views (Subsection 4.4.1) and Wang and Zhang's equilibrium views (Subsection 4.4.2)), as well as with previous approaches that have tried to generalise or to refine **E-S** and compare them all with our approach.

To the best of our knowledge, it was Chen who achieved to embed **E-S** into an epistemic modal logic with a kind of minimal model reasoning about epistemic concepts knowledge and belief Chen [1997]. Therefore, his approach significantly differs from ours because our approach may be viewed a kind of intuitionistic modal logic with the nonduality of our epistemic operators. Chen has recognised the close resemblance between the notion of only knowing in the logic of only knowing (**OK**) (Levesque [1990]) and the notion of world views of **E-S**. However, **OK** is not strong enough to cover the notion of world views. Therefore, he has improved its syntax and semantics, and paved the way for the generalised logic of only knowing (**G-OK**) (Chen [1997]) which also contains **OK** as a subset. As a result, he has also kept the important features of **OK**. What are these exactly? **OK** is a modal logic that helps us formalise an agent's introspective reasoning and answer epistemic questions concerning databases. It has been shown in time that **OK** has some relations with the logic of minimal belief and negation as failure (**MB-NF**) (Lifschitz [1994]) by Lifshitz and extended logic programs (Chen [1993, 1994]), so has been proved that **OK** provides a general logical framework for non-monotoning reasoning. **G-OK** further enhances the expressive power of **OK** with

more desirable qualities so as to include epistemic specifications. It is worth to mentioning that different from standard propositional modal logic, for instance **S5** (see Blackburn et al. [2001a]; Hughes and Cresswell [2012]), **G-OK** succeeds in creating a kind of minimal model reasoning about epistemic concepts (i.e, knowledge and belief). In this respect, its semantics shares a similar spirit of Gelfond's world view semantics. Eventually, Chen has materialised this embedding. However, it would be fair to say that his logic has syntactically and semantically a complex nature due to the display of four different modal operators. As a result, it is a bit difficult to observe the intuition that lays behind.

Wang and Zhang generalised **E-S** into an epistemic extension of equilibrium logic (Wang and Zhang [2005]). that precisely captures reasoning with epistemic specifications. Moreover, this extension also allows for generalisations to arbitrary programs with nested epistemic rules. However, although the general structures meet, our approaches have serious differences in essence which are discussed in Subsection 4.4.2 in detail.

More recently, Truszczyński has generalised a refinement of **E-S** (Truszczyński [2011]), but his formalism does not deal with **HT** and equilibrium logic, and so it is not directly relevant for our work. Moreover, having proposed a model $\{\emptyset\}$ for the formula $(p \vee \neg p) \rightarrow p$, he leaves apart from the main approach because this formula has no equilibrium models.

All these approaches deal with the former version of **E-S** discussed in Baral and Gelfond [1994]; Gelfond [1991, 1994]. A well-known distinctive example between Gelfond's former and recent versions of **E-S** is $T = \{p \leftarrow \mathsf{K}\, p\}$: while the first version gives two world views $\{\emptyset\}$ and $\{\{p\}\}$, the second eliminates the unintended world view $\{\{p\}\}$.

4.4.1 AEEMs versus world views

We first translate $\mathcal{L}_{\textbf{E-S}}$ to $\mathcal{L}_{\textbf{E-HT}}$ via a mapping tr, which introduces a fresh variable $\widetilde{p}$ for each $p \in \mathbb{P}$. We call $\widetilde{p}$ the 'opposite' of p. The translation of an epistemic specification T involves an EHT theory that we call *constraints of* T, $\mathsf{Cons}(T)$, ensuring that when $\widetilde{p}$ occurs in $\mathsf{tr}(T)$ then p and $\widetilde{p}$ cannot be true at the same time. It is defined as:

$$\mathsf{Cons}(T) = \Big(\bigwedge_{p \in \mathbb{P}_T} (p \wedge \widetilde{p}) \rightarrow \bot \Big).$$

The rest is straightforward: we replace '$\leftarrow$', '*or*', '*,*' and '*not*' respectively by '$\rightarrow$', '$\vee$', '$\wedge$' and '$\neg$'. So, an ES rule like '$p \, or \sim q \leftarrow r \, , \, not \, s$' is translated into '$r \wedge \neg s \rightarrow p \vee \widetilde{q}$ ' accompanied by $q \wedge \widetilde{q} \rightarrow \bot$. Note that the other conjuncts of Cons are redundant here.

We now give the formal translation, and start by translating literals.

For an objective literal in $\mathbb{O}$-*Lit* we define:

$$\begin{array}{lcl} \mathsf{tr}(p) & = & p; \\ \mathsf{tr}(\sim p) & = & \widetilde{p}. \end{array}$$

This allows us to translate Gelfond's simple theories S: we define $\mathsf{tr}(S) = \{\mathsf{tr}(l) \ : \ l \in S\}$.

We suppose that the M operator has been replaced as described in Section 1.4.2, so for a subjective literal in $\mathbb{S}$-*Lit* it suffices to define:

$$\begin{array}{lcl} \mathsf{tr}(\mathsf{K}\, l) & = & \mathsf{K}\, \mathsf{tr}(l); \\ \mathsf{tr}(\mathsf{K}\, not\, l) & = & \mathsf{K}\, \neg \mathsf{tr}(l); \\ \mathsf{tr}(\sim \mathsf{K}\, l) & = & \neg \mathsf{K}\, \mathsf{tr}(l); \\ \mathsf{tr}(\sim \mathsf{K}\, not\, l) & = & \neg \mathsf{K}\, \neg \mathsf{tr}(l). \end{array}$$

for $l \in \mathbb{O}$-*Lit*. However, given the equivalences in (1.22) of Section 1.4 in Introduction, subjective literals can also be translated in a slightly different way:

$$\begin{array}{lcllcl} \mathsf{tr}(\sim \mathsf{M}\ not\, l) & = & \mathsf{tr}(\mathsf{K}\, l) & = & \mathsf{K}\, \mathsf{tr}(l); \\ \mathsf{tr}(\sim \mathsf{M}\, l) & = & \mathsf{tr}(\mathsf{K}\ not\, l) & = & \mathsf{K}\, \neg \mathsf{tr}(l); \\ \mathsf{tr}(\mathsf{M}\ not\, l) & = & \mathsf{tr}(\sim \mathsf{K}\, l) & = & \neg \mathsf{K}\, \mathsf{tr}(l); \\ \mathsf{tr}(\mathsf{M}\, l) & = & \mathsf{tr}(\sim \mathsf{K}\ not\, l) & = & \hat{\mathsf{K}}\, \mathsf{tr}(l). \end{array}$$

The subtlety of the translation are those of $\mathsf{K}\, l$ and $\mathsf{M}\, l$ because we have two alternatives in each case since K and $\hat{\mathsf{K}}$ are not dual. However, while $\neg \hat{\mathsf{K}}\, \neg p$ and $\neg \mathsf{K}\, \neg p$ have no EEM (see Subsection 4.2), $\mathsf{K}\, p$ and $\hat{\mathsf{K}}\, p$ have EEMs (and therefore also AEEMs) (see Subsection 4.3.1). Note that we generalise **E-S** to nested epistemic logic programs (NELPs), and this translation gives more intuitive results in general. $\neg \mathsf{K}\, \neg p$ has some further strange behaviours: while $\{\emptyset, \underline{\{p\}}\}$ is an AEEM of $p \wedge \neg \mathsf{K}\, \neg p$ and $\neg p \wedge \neg \mathsf{K}\, \neg p$ has none, the AEEMs of $p \wedge \hat{\mathsf{K}}\, p$ and $\neg p \wedge \hat{\mathsf{K}}\, p$ are respectively $\{\emptyset, \underline{\{p\}}\}$ and $\{\underline{\emptyset}, \{p\}\}$ which are more intuitive. Finally, also note that $\mathsf{K}\, \varphi \rightarrow \neg \hat{\mathsf{K}}\, \neg \varphi$ and $\hat{\mathsf{K}}\, \varphi \rightarrow \neg \mathsf{K}\, \neg \varphi$ are EHT valid (see Subsection 4.1.4). The rest is safe: $\neg \hat{\mathsf{K}}\, \varphi$ and $\mathsf{K}\, \neg \varphi$ are EHT equivalent, so are $\hat{\mathsf{K}}\, \neg \varphi$ and $\neg \mathsf{K}\, \varphi$ (see Proposition 4.4). Hence we could as well write $\hat{\mathsf{K}}\, \neg$ instead of $\neg \mathsf{K}$ in line 3 above.

After the translation of the primitives are given, we now translate epistemic rules:

$$\begin{array}{l} \mathsf{tr}\Big(l_1 \ or \ \ldots \ or \ l_k \leftarrow g_{k+1}, \ldots, g_m, \ not \ l_{m+1}, \ldots, \ not \ l_n\Big) = \\ \Big(\mathsf{tr}(g_{k+1}) \wedge \ldots \wedge \mathsf{tr}(g_m) \wedge \neg \mathsf{tr}(l_{m+1}) \wedge \ldots \wedge \neg \mathsf{tr}(l_n)\Big) \rightarrow \Big(\mathsf{tr}(l_1) \vee \ldots \vee \mathsf{tr}(l_k)\Big). \end{array}$$

It will be useful to consider a typical rule of the form

$$\rho = \alpha \leftarrow \mathsf{K}\, l_1, \sim \mathsf{K}\, l_2, \mathsf{K}\, not\, l_3, \sim \mathsf{K}\, not\, l_4, \beta$$

where α and β are boolean formulas. Its translation is given as:

$$\mathsf{tr}(\rho) = \Big(\mathsf{K}\,\mathsf{tr}(l_1) \wedge \neg\mathsf{K}\,\mathsf{tr}(l_2) \wedge \mathsf{K}\,\neg\mathsf{tr}(l_3) \wedge \neg\mathsf{K}\,\neg\mathsf{tr}(l_4) \wedge \mathsf{tr}(\beta)\Big) \rightarrow \mathsf{tr}(\alpha).$$

Finally, we translate an epistemic specification T by $\mathsf{tr}(T) = \mathsf{K}\,\Big(\bigwedge_{\rho\in T}\mathsf{tr}(\rho)\Big)$. Here are some very basic examples:

$$\begin{array}{lcl}
\mathsf{tr}(\leftarrow p) & = & \mathsf{tr}(p) \rightarrow \mathsf{tr}(\bot) = p \rightarrow \bot = \neg p \\
\mathsf{tr}(\leftarrow \sim p) & = & \mathsf{tr}(\sim p) \rightarrow \mathsf{tr}(\bot) = \widetilde{p} \rightarrow \bot = \neg\widetilde{p} \\
\mathsf{tr}(p \textit{ or } \sim p \leftarrow) & = & \mathsf{tr}(\top) \rightarrow \Big(\mathsf{tr}(p) \vee \mathsf{tr}(\sim p)\Big) = \top \rightarrow (p \vee \widetilde{p}) \\
\mathsf{tr}(\leftarrow p, \sim p) & = & \Big(\mathsf{tr}(p) \wedge \mathsf{tr}(\sim p)\Big) \rightarrow \mathsf{tr}(\bot) = (p \wedge \widetilde{p}) \rightarrow \bot = \neg(p \wedge \widetilde{p}).
\end{array}$$

In the third example, $\top \rightarrow (p \vee \widetilde{p})$ is equivalent to $p \vee \widetilde{p}$.

One of the main differences between AEEMs and world views is that the latter handle each ES rule ρ like $\mathsf{K}\,\rho$: it does not require pointed S5 models. As a result of this, non-pointed world view has a very special character: any belief set of the collection describes a possible world obtained from an epistemic specification T that stands alone, so it in a way ignores a collection accompanying these possible worlds in general, and may lose some information. In contrast, in our approach we consider some identified belief set (aka actual world) together with a collection. This failure may not appear as a serious problem with **E-S** since K and M can only appear in the body of its rules. It can still be discussed if $p \leftarrow \mathsf{M}\ \textit{not}\, p$ should have a world view. According to Gelfond and Kahl's approaches, it does not have any, whereas this rule when viewed an EHT formula has an epistemic equilibrium model, namely $\{\emptyset, \underline{\{p\}}\}$. Moreover, in a possible generalisation to NELPs, for instance, it seems there would be no world view candidate to $\textit{not}\, p\,, \mathsf{M}\, p \leftarrow$ whereas $\neg p \wedge \hat{\mathsf{K}}\, p$ has an AEEM $\{\underline{\emptyset}, \{p\}\}$ which is intuitive.

The examples (1.23) and (1.24) of Section 1.4.2 in Introduction show that the approaches by Gelfond, Kahl and us are all different in the case of cyclic dependences: the AEEMs of $\hat{\mathsf{K}}\, p \rightarrow p$ and $(p \vee q) \wedge (\hat{\mathsf{K}}\, q \rightarrow p)$ are respectively $\{\emptyset\}$ and $\{\{p\}\}$. Another example is the epistemic specification

$$T = \Big\{\ p \textit{ or } q\ \ ,\ \ q \leftarrow \sim\mathsf{K}\, p\ \Big\}.$$

Gelfond's world views of T are $\{\{p\}\}$ and $\{\{q\}\}$, which matches the unique AEEM of $(\neg q \rightarrow p) \wedge (\neg K p \rightarrow q)$ is $\{\{p\}\}$. In contrast, Kahl only obtains the second world view. Finally, we consider an epistemic specification

$$T' = \Big\{\ p\ \ \leftarrow\ \ q\ \ ,\ \ q \leftarrow \sim\mathsf{M}\, p\ \Big\}.$$

According to Kahl's semantics, while T' has a unique world view $\{\{p, q\}\}$, it has two world views according to Gelfond's semantics, namely $\{\emptyset\}$ and $\{\{p, q\}\}$. We get the epistemic equilibrium model $\{\emptyset\}$ when we translate $\mathsf{M}\, p$ as $\hat{\mathsf{K}}\, p$, and we get $\{\emptyset\}$ and $\{\{p, q\}\}$ when we translate it by $\neg\mathsf{K}\,\neg p$.

4.4.2 AEEMs versus equilibrium views

In 2005, Wang and Zhang described an epistemic extension of equilibrium logic into which they were able to embed Gelfond's first version of **E-S** (Wang and Zhang [2005]). This extension also gives semantics to nested epistemic logic programs (NELPs). The underlying language is that of **HT** with an additional formation rule:

- if φ is a formula then so are $\mathsf{K}\,\varphi$ and $\mathsf{M}\,\varphi$.

Then, they also add strong negation into the language and discuss the resulting formalism briefly.

The semantics is given by WZ-EHT models $(\mathcal{A}, H, T)$ in which

- $\mathcal{A} \subseteq 2^{\mathbb{P}}$ is a nonempty collection of valuations;
- (H, T) is an HT model.

Note that H and T are not necessarily contained in $\mathcal{A}$. Given $\mathcal{A} \subseteq 2^{\mathbb{P}}$, they define the collection

$$\mathit{coll}(\mathcal{A}) = \{(H, T) \ : \ H, T \in \mathcal{A} \text{ such that } H \subseteq T\}.$$

Then the additional truth conditions can be defined as follows:

$$(\mathcal{A}, H, T) \models_{\textbf{E-HT}} \mathsf{K}\,\varphi \text{ if } \mathit{coll}(\mathcal{A}), A \models_{\textbf{E-HT}} \varphi \text{ for all } A \in \mathit{coll}(\mathcal{A});$$
$$(\mathcal{A}, H, T) \models_{\textbf{E-HT}} \mathsf{M}\,\varphi \text{ if } \mathit{coll}(\mathcal{A}), A \models_{\textbf{E-HT}} \varphi \text{ for some } A \in \mathit{coll}(\mathcal{A}).$$

WZ-EEMs for an EHT theory Φ are total EHT models $(\mathcal{A}, T, T)$ of Φ such that there is no EHT model $(\mathcal{A}, H, T)$ of Φ with $H \subset T$. Finally, Wang and Zhang capture the world views of an epistemic specification T by *equilibrium views* of Φ_T (where Φ_T is the epistemic HT theory corresponding to T), which are defined to be the maximal collections $\mathcal{A} \subset 2^{\mathbb{P}}$ satisfying the following fixed point equation:

$$\mathcal{A} = \{T \ : \ (\mathcal{A}, T, T) \text{ is an equilibrium model of } \Phi_T\}.$$

At first sight, their unifying framework for both nested logic programs and epistemic specifications looks fairly strong, and we accept that what we have suggested in this chapter is slightly similar to their work. However, we also dare to say that the epistemic **HT** logic they have proposed can just be considered a starting point of our work because the approach in Wang and Zhang [2005] requires a better-designed syntax character, and has several drawbacks that are due to the design of the semantics. Basically, it has much less epistemic HT models than in our approach: only the actual world is an HT pair, while the set of accessible

worlds is a set of valuations as in classical **S5**. Sets of HT pairs are generated from that set by taking all possible pairs satisfying the heredity. It is much less obvious to extend the semantics of Wang and Zhang [2005] beyond single-agent epistemic logic, as compared to our semantics. Moreover, the resulting epistemic **HT** logic of Wang and Zhang [2005] has some strange properties:

1. $\mathsf{M}\,p \leftrightarrow \neg\mathsf{K}\,\neg p$ is valid, i.e., K and M are dual.
2. $\mathsf{M}\,\neg\neg p \rightarrow \mathsf{M}\,p$ is valid while $\mathsf{K}\,\neg\neg p \rightarrow \mathsf{K}\,p$ is not.
3. $\mathsf{M}\,p \wedge \mathsf{M}\,\neg p$ has an epistemic equilibrium model in our semantics, viz. $\{\emptyset, \{p\}\}$, which is also the unique autoepistemic model; in contrast, according to Wang and Zhang [2005] it has no equilibrium view.

(1) and (2) disqualify the epistemic **HT** logic of Wang and Zhang [2005] as an intuitionistic modal logic. Due to (3), the extension of epistemic specifications beyond Gelfond has unintuitive properties. On the other hand, we combine both **S5** logic and **HT** logic into the resulting logic **E-HT**, revealing their natures apparently. To sum up, **E-HT** unveils the modal logic perspective, yet keeping the **HT** logic character. Obviously, with the syntactical power to express the notions of both epistemic specification and nested logic program, as well as a simple and a neat semantics, epistemic equilibrium logic is a fairly attractive candidate to provide a general framework for various forms of non-monotonic reasoning.

Finally, we summarise the concise relationship between our approaches:

1. Inclusion of satisfiability: each satisfiable $\varphi \in \mathcal{L}_{\textbf{E-HT}}$ has also a WZ-EHT model; the other way round, for instance $\mathsf{K}\,p \wedge \neg p$ has a WZ-EHT model but no EHT model.
2. The minimality is totally different: both $\mathsf{K}\,p \wedge \neg p$ and $\mathsf{K}\,\neg\neg p$ have WZ-EEMs but no EEMs in our sense; the other way round, $\mathsf{K}\,p \wedge \neg\neg p$ has an EEM in our sense, but no WZ-EEM.
3. Maximal ignorance is different as well: $\mathsf{K}\,p$ has an AEEM, but no WZ-equilibrium view. $\hat{\mathsf{K}}\,p$, $\hat{\mathsf{K}}\,p \wedge \neg p$, $(p \vee q) \wedge (\hat{\mathsf{K}}\,q \rightarrow p)$ and $p \vee \mathsf{K}\,(p \vee \neg p)$ are other examples. $(\hat{\mathsf{K}}\,\neg p \vee \hat{\mathsf{K}}\,\neg\neg p) \wedge (\hat{\mathsf{K}}\,p \rightarrow (p \vee \neg p))$ has a WZ-equilibrium view, but no AEEM.

4.5 Conclusion and future work

In this chapter, we have designed a simple, neat and monotonic intuitionistic modal logic **E-HT**: we have added two modal operators K and $\hat{\mathsf{K}}$ to the original language of **HT** that are interpreted in a straightforward combination of **S5** and **HT**. Based

on this logic we have defined first epistemic equilibrium models and then autoepistemic equilibrium models. The latter are powerful enough to provide a new logical semantics not only for **E-S**, but also for programs with nested expressions containing belief operators. We have provided strong equivalence characterisations and have compared our semantics with the existing semantics for **E-S**. Our semantics performs more or less like Gelfond's if there are cyclic dependences, such as in $p \leftarrow \mathsf{M}\, p$. It seems to us that is not clear which semantics better matches our intuitions in such cases and that the question of a fully satisfactory semantics of **E-S** is therefore still open. However, we believe that **E-HT** and the EMs built on it provide a good starting point for further extensions of **ASP** by modal concepts.

Given this first step in an epistemic equilibrium logic, in future work we plan to extend our approach in four steps: (1) the first step is further to base our approach on N_5 and investigate how the underlying epistemic version of N_5 can be axiomatised, and how appropriate proof systems can be built. (2) Then we can generalise the single-agent epistemic equilibrium logic to a multi-agent version. (3) We also plan to capture kind of world view notion in a dynamic extension of **HT** logic that is able to cover the single-agent epistemic equilibrium logic as well. This dynamic extension should let us update a collection of HT models, different from an update of a single HT model that has been already proposed in Chapter 3. (4) Finally, we would like to propose a dynamic epistemic extension of **HT** logic in order to talk about belief change. We plan to do so by combining our approach with the approach of Fariñas del Cerro et al. [2013]. As mentioned also in Chapter 3, Fariñas del Cerro et al. [2013] shows that equilibrium models and their updates can be analysed in terms of **D-HT** of atomic change of equilibrium models which is strongly related to **DL-PA**. The resulting extension also allows us to capture all previous approaches that are introduced up to now.

Chapter 5

Summary and Future Research

ASP has emerged as a viable tool for Knowledge Representation and Reasoning. It offers efficient and versatile off-the-shelf solving technology, and provides an expanding functionality and ease of use as a rapid application development tool. Moreover, **ASP** has a continuously growing range of applications:

$$\textbf{ASP} = \text{DB} + \text{LP} + \text{KR} + \text{SAT}$$

We plan to extend our approach in two ways: (1) first, we plan to continue extensions of the original language of **HT** logic with modal operators allowing to talk about action providing thus a more comprehensive framework for extensions of answer set programming. Another natural continuation of our work would be to compare existing temporal extensions of equilibrium logic by Cabalar *et al.* (Aguado et al. [2008]; Cabalar and Demri [2011]) with our modal extensions of equilibrium logic. The very next step is to seek for what further can be done to materialize an update of a collection of HT models with dynamic operators. (2)Our second research avenue is to capture other nonmonotonic logics such as the nonmonotonic extension of **S4F** Pearce and Uridia [2011] as well as default logic.

Appendix A

All proofs

Proofs of theorems, propositions and lemmas that appear in this thesis work are included below.

A.1 Proofs of Chapter 1

Lemma A.1 *Given an* $\mathcal{L}_{\mathbf{HT}}$ *formula* φ *and a propositional variable* $q \in \mathbb{P}$ *such that* $q \notin \mathbb{P}_\varphi$*, we have*

$$H, T \models_{\mathbf{HT}} \varphi \quad \textit{iff} \quad H, T \cup \{q\} \models_{\mathbf{HT}} \varphi \quad \textit{iff} \quad H \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \varphi.$$

Proof. The proof is by structural induction (i.e., on the form of φ): the case where φ is a propositional variable is easy, and the case where φ equals $\bot$ is immediate. As for the first two Boolean cases, they are straightforward, so we just give the proof of the implication case. Let $\varphi = \psi_1 \to \psi_2$ for some $\psi_1, \psi_2 \in \mathcal{L}_{\mathbf{HT}}$ and let $q \in \mathbb{P} \setminus \mathbb{P}_\varphi$. Then $q \notin \mathbb{P}_{\psi_i}$ for $i = 1, 2$. We get the result through the following sequence of equivalent steps:

1. $H, T \models_{\mathbf{HT}} \psi_1 \to \psi_2$
2. $H, T \models_{\mathbf{HT}} \psi_1 \supset \psi_2$ and $T, T \models_{\mathbf{HT}} \psi_1 \supset \psi_2$
3. $H, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \supset \psi_2$ and $T \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \supset \psi_2$ (from 2 by I.H.)
4. $H, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \to \psi_2$
5. $H \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \supset \psi_2$ and $T \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \supset \psi_2$ (from 2 by I.H.)
6. $H \cup \{q\}, T \cup \{q\} \models_{\mathbf{HT}} \psi_1 \to \psi_2$.

Hence, the result follows from the equivalence of (1), (4) and (6). q.e.d.

A.2 Proofs of Chapter 2

Proposition A.1 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model.*

1. *For every w, u, if $w\mathcal{T}u$ then $V_w \subseteq V_u$.*
2. *For every w, if $\mathcal{T}(w) \setminus \{w\}$ is empty, then the set $\{V_u : w\mathcal{S}u\}$ equals either $\{V : V \subseteq V_w\}$ or $\{V : V \subset V_w\}$.*

Proof. The first property can be proved from wmconv($\mathcal{T}, \mathcal{S}$) and heredity($\mathcal{S}$). As for the second property, it is due to neg($\mathcal{S}, \mathcal{T}$), refl($\mathcal{T}$) and again heredity($\mathcal{S}$).

1. For $w, u \in W$, let $w\mathcal{T}u$. If $w = u$ then the result trivially follows. Suppose $w \neq u$. Hence, by wmconv($\mathcal{T}, \mathcal{S}$) we have $u\mathcal{S}w$ as well. Finally, heredity($\mathcal{S}$) guarantees $V_w \subseteq V_u$.

2. We first assume that $\mathcal{T}(w) \setminus w = \emptyset$. Then using also refl($\mathcal{T}$), we get $\mathcal{T}(w) = \{w\}$. Now, we consider two cases.
   Case 1: let $V_w \neq \emptyset$. Then, by neg($\mathcal{S}, \mathcal{T}$) we have: for every $P \subseteq V_w$ such that $P \neq \emptyset$, there is $v \in W$ with $w\mathcal{S}v$ and $V_v = V_w \setminus P$. Thus, $\{V_u : w\mathcal{S}u\} = \mathcal{P}(V_w) \setminus \{V_w\}$ which equals nothing, but $\{V : V \subset V_w\}$.
   Case 2: now let $V_w = \emptyset$, so here we cannot use neg($\mathcal{S}, \mathcal{T}$) anymore, and no other constraint of **MEM** frames says something whether or not $\mathcal{S}(w)$ is empty. Therefore, we need to go over both cases:

   - Let $\mathcal{S}(w) = \emptyset$. Then $\{V_u : w\mathcal{S}u\} = \emptyset = \{V : V \subset V_w\}$.
   - Let $\mathcal{S}(w) \neq \emptyset$. Then there is at least one u such that $w\mathcal{S}u$. Moreover, by heredity($\mathcal{S}$), for every v such that $w\mathcal{S}v$, we have $V_v = \emptyset$. Hence, $V_u = \emptyset$. Thus, $\{V_u : w\mathcal{S}u\} = \{\emptyset\} = \{V : V \subseteq V_w\}$.

q.e.d.

Proposition A.2 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model, and φ be an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula. Let the valuation V^φ be defined as follows:*

$$V^\varphi_w = V_w \cap \mathbb{P}_\varphi, \text{ for every } w \in W.$$

Then $M^\varphi = (W, \mathcal{T}, \mathcal{S}, V^\varphi)$ is also a **MEM** *model. Moreover, for every $w \in W$,*

$$M, w \models \varphi \text{ if and only if } M^\chi, w \models \varphi,$$

where φ is a subformula of χ.

PROOF. First, let us show that M^φ is a **MEM** model for a $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula φ, so we need to check that all constraints hold in M^φ. The frame constraints are only about the accessibility relations and are clearly preserved because we just modify the valuation. As for the constraints involving the valuation, the model M^φ satisfies heredity($\mathcal{S}$) constraint: suppose $w\mathcal{S}u$; as M satisfies heredity($\mathcal{S}$) we have $V_u \subseteq V_w$; hence $V_u^\varphi \subseteq V_w^\varphi$ as well. Finally, the model M^φ also satisfies the constraint neg($\mathcal{S}, \mathcal{T}$): for every $w \in W$, by the constraints refl($\mathcal{T}$) and alt$_2$($\mathcal{T}$) there exists either one or two u such that $w\mathcal{T}u$; in the former case, $u = w$ whereas in the latter, we choose u different from w; for such u's, let $V_u^\varphi = V_u \cap \mathbb{P}_\varphi$ and $P \subseteq V_u^\varphi$ be non-empty; then since M satisfies the neg($\mathcal{S}, \mathcal{T}$) constraint there is v with $u\mathcal{S}v$ satisfying $V_v = V_u \setminus P$; clearly, for that v we also have $V_v^\varphi = V_u^\varphi \setminus P$ since $V_v^\varphi = V_v \cap \mathbb{P}_\varphi = (V_u \setminus P) \cap \mathbb{P}_\varphi = (V_u \cap \mathbb{P}_\varphi) \setminus P = V_u^\varphi \setminus P$.
Second, we prove that if φ is a subformula of χ then $M, w \models \varphi$ if and only if $M^\chi, w \models \varphi$, by induction on the form of φ. The base cases are routine. The Boolean case is easy, yet we give the proof. Let $\gamma = \varphi \supset \psi$.

$$\begin{array}{lll} M, w \models \varphi \supset \psi & \text{iff} & M, w \not\models \varphi \text{ or } M, w \models \psi \\ & \text{iff} & M^\chi, w \not\models \varphi \text{ or } M^\chi, w \models \psi \text{ for } \varphi \text{ and } \psi, \text{ subformulas of } \chi \\ & & \qquad\qquad\qquad\qquad\qquad\qquad \text{(by I.H.)} \\ & \text{iff} & M^\xi, w \models \varphi \supset \psi \text{ where } \gamma \text{ is a subformula of } \xi. \end{array}$$

As for the modalities, we only give the proof for the case where φ is of the form $[\mathrm{T}]\psi$, yet the case $[\mathrm{S}]\psi$ is similar. We have:

$$\begin{array}{lll} M, w \models [\mathrm{T}]\psi & \text{iff} & M, u \models \psi \text{ for every } u \text{ such that } w\mathcal{T}u \\ & \text{iff} & M^{[\mathrm{T}]\psi}, u \models \psi \text{ for every } u \text{ such that } w\mathcal{T}u \qquad \text{(by I.H.)} \\ & \text{iff} & M^\varphi, w \models [\mathrm{T}]\psi. \end{array}$$

q.e.d.

Proposition A.3 *The schema Heredity*([T])*, i.e.,* $\varphi^+ \supset [\mathrm{T}]\varphi^+$*, for* φ^+ *a positive Boolean formula, is provable.*

PROOF.

1. $\langle \mathrm{S} \rangle \varphi^+ \supset \varphi^+$ (Heredity([S]))
2. $\varphi^+ \supset [\mathrm{T}](\varphi^+ \vee \langle \mathrm{S} \rangle \varphi^+)$ (WMConv([T], [S]))
3. $\varphi^+ \supset [\mathrm{T}]\varphi^+$ (from 1 and 2 by K[S]).

q.e.d.

Proposition A.4 *The schema* 2([T])*, i.e.,* $\langle \mathrm{T}\rangle[\mathrm{T}]\varphi \supset [\mathrm{T}]\langle \mathrm{T}\rangle\varphi$ *is provable.*

Proof.

1. $\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \supset \langle \mathrm{T}\rangle{\sim}\varphi$ (from T([T]) by K([T]))
2. $\langle \mathrm{T}\rangle[\mathrm{T}]\varphi \supset \langle \mathrm{T}\rangle\big([\mathrm{T}]\varphi \wedge \varphi\big)$ (from T([T]) by K([T]))
3. $[\mathrm{T}]\varphi \vee [\mathrm{T}]\big(\varphi \supset \langle \mathrm{T}\rangle{\sim}\varphi\big) \vee [\mathrm{T}]\big((\varphi \wedge \langle \mathrm{T}\rangle{\sim}\varphi) \supset \bot\big)$ (Alt_2([T]))
4. $\big(\langle \mathrm{T}\rangle{\sim}\varphi \wedge \langle \mathrm{T}\rangle(\varphi \wedge [\mathrm{T}]\varphi)\big) \supset [\mathrm{T}]\big(\varphi \supset [\mathrm{T}]\varphi\big)$ (from 3 by K([T]))
5. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset [\mathrm{T}]\big(\varphi \supset [\mathrm{T}]\varphi\big)$ (from 1, 2 and 4 by K([T]))
6. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset [\mathrm{T}]\big({\sim}\varphi \supset [\mathrm{T}]{\sim}\varphi\big)$ (from 5 by K([T]))
7. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset [\mathrm{T}]\big([\mathrm{T}]\varphi \vee [\mathrm{T}]{\sim}\varphi\big)$ (from 5 and 6 by K([T]))
8. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset \big([\mathrm{T}][\mathrm{T}]\varphi \vee [\mathrm{T}][\mathrm{T}]{\sim}\varphi\big)$ (from 7 by T([T]) & 4([T]))
9. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge [\mathrm{T}][\mathrm{T}]\varphi\big) \supset \langle \mathrm{T}\rangle[\mathrm{T}]\bot$ (by K([T]))
10. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge [\mathrm{T}][\mathrm{T}]{\sim}\varphi\big) \supset [\mathrm{T}][\mathrm{T}]{\sim}\varphi$ (by K([T]))
11. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge ([\mathrm{T}][\mathrm{T}]\varphi \vee [\mathrm{T}][\mathrm{T}]{\sim}\varphi)\big) \supset \big(\langle \mathrm{T}\rangle[\mathrm{T}]\bot \vee [\mathrm{T}][\mathrm{T}]{\sim}\varphi\big)$ (from 9 & 10)
12. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]\varphi \wedge ([\mathrm{T}][\mathrm{T}]\varphi \vee [\mathrm{T}][\mathrm{T}]{\sim}\varphi)\big) \supset \big(\langle \mathrm{T}\rangle[\mathrm{T}]\bot \vee [\mathrm{T}][\mathrm{T}]\varphi\big)$ (from 11 - K([T]))
13. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi \wedge ([\mathrm{T}][\mathrm{T}]\varphi \vee [\mathrm{T}][\mathrm{T}]{\sim}\varphi)\big) \supset \big(\langle \mathrm{T}\rangle[\mathrm{T}]\bot \vee ([\mathrm{T}][\mathrm{T}]\varphi \wedge [\mathrm{T}][\mathrm{T}]{\sim}\varphi)\big)$ (from 11 and 12 by K([T]))
14. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset \big(\langle \mathrm{T}\rangle[\mathrm{T}]\bot \vee ([\mathrm{T}][\mathrm{T}]\varphi \wedge [\mathrm{T}][\mathrm{T}]{\sim}\varphi)\big)$ (from 8 and 13)
15. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset \big(\langle \mathrm{T}\rangle[\mathrm{T}]\bot \vee \bot\big)$ (from 14 by K([T]) and T([T]))
16. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]\varphi\big) \supset \langle \mathrm{T}\rangle\bot$ (from 15 by T([T]) and K([T]))
17. $\big(\langle \mathrm{T}\rangle[\mathrm{T}]\varphi \wedge \langle \mathrm{T}\rangle[\mathrm{T}]{\sim}\varphi\big) \supset \bot$ (from 16 by K([T]))
18. $\langle \mathrm{T}\rangle[\mathrm{T}]\varphi \supset [\mathrm{T}]\langle \mathrm{T}\rangle\varphi$ (from 17 by K([T])).

q.e.d.

Lemma A.2 *The following formula schema is provable:*

$$Neg'([\mathrm{S}],[\mathrm{T}]) \qquad \langle \mathrm{T}\rangle \Big((\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle \Big((\bigwedge_{p\in P} {\sim}p) \wedge (\bigwedge_{q\in Q} q)\Big)$$

for $P, Q \subseteq \mathbb{P}$ finite, $P \neq \emptyset$, and $P \cap Q = \emptyset$.

Proof. $\mathrm{Neg}'([\mathrm{S}],[\mathrm{T}])$ can be proved using the axiom schema $\mathrm{Neg}([\mathrm{S}],[\mathrm{T}])$ by standard modal logic principles, i.e., by $\mathbf{K}([\mathrm{T}])$. Suppose P and Q are finite subsets of $\mathbb{P}$ such that $P \neq \emptyset$ and $P \cap Q = \emptyset$. The implication

$$\Big((\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \Big((\bigvee_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big)$$

is valid in classical propositional logic. Then $\mathrm{Neg}'([\mathrm{S}],[\mathrm{T}])$ follows through the argument below:

1. $(\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q) \supset (\bigvee_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)$ (tautology)
2. $\langle \mathrm{T}\rangle\Big((\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \langle \mathrm{T}\rangle\Big((\bigvee_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big)$ (from 1 by K[T])
3. $\langle \mathrm{T}\rangle\Big((\bigvee_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\Big((\sim \bigvee_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big)$ ($\mathrm{Neg}([\mathrm{S}],[\mathrm{T}])$)
4. $\langle \mathrm{T}\rangle\Big((\bigwedge_{p\in P} p) \wedge (\bigwedge_{q\in Q} q)\Big) \supset \langle \mathrm{T}\rangle\langle \mathrm{S}\rangle\Big((\bigwedge_{p\in P} {\sim}p) \wedge (\bigwedge_{q\in Q} q)\Big)$ (2 & 3 - K[T]).

q.e.d.

Theorem A.1 *Let φ be an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula. Then φ is* **MEM** *valid if and only if φ is provable from the axioms and the inference rules of* **MEM**.

Proof. Soundness is proved as usual. We just consider the proof of axiom schema $\mathrm{Neg}([\mathrm{S}],[\mathrm{T}])$. Let φ^+ be a positive Boolean formula such that $\mathbb{P}_{\varphi^+} \cap \mathbb{P}_{\psi} = \emptyset$. Suppose

$$M, w \models \langle \mathrm{T}\rangle(\varphi^+ \wedge \psi) \qquad (*).$$

Put φ^+ in conjunctive normal form (CNF), and let $\kappa = (\bigvee P)$ be a clause of this CNF, for some $P \subseteq \mathbb{P}_{\varphi^+}$. Observe that $P \neq \emptyset$ by the definition of positive Boolean formulas and CNF. Now, we need to consider two cases (according to Remark 2.2). Case (1): let $\mathcal{T}(w) \setminus \{w\} = \emptyset$. Hence $\mathcal{T}(w) = \{w\}$ by the reflexivity of $\mathcal{T}$. Then

from $(*)$ we obtain that $M, w \models \varphi^+ \wedge \psi$ $(**)$. Moreover, $M, w \models \varphi^+$ implies that $V_w \neq \emptyset$. In addition, non-emptiness of V_w yields that w is not a singleton point (because singleton points always have an empty valuation; otherwise that would contradict $\mathrm{neg}(\mathcal{S}, \mathcal{T})$). Now, take $P_w = P \cap V_w$. We have $P_w \neq \emptyset$ because $M, w \models \kappa$ (since we have $M, w \models \varphi^+$). As M satisfies the constraint $\mathrm{neg}(\mathcal{S}, \mathcal{T})$, there exists u with $w\mathcal{T}u$, but since $\mathcal{T}(w) = \{w\}$ we have $u = w$. Since $V_w \neq \emptyset$, according to the negatable constraint, for non-empty $P_w \subseteq V_w$, there is v such that $w\mathcal{S}v$ and $V_v = V_w \setminus P_w$. Since $P_w \cap V_v = \emptyset$, we also have $P \cap V_v = \emptyset$ (because $V_v \subseteq V_w$, but $P \setminus P_w \not\subseteq V_w$). Hence, $M, v \not\models \kappa$. As a result, $M, v \not\models \varphi^+$ either. So $M, v \models \sim\varphi^+$. In addition, $M, v \models \psi$ because $M, w \models \psi$ by $(**)$ and $V_w \cap \mathbb{P}_\psi = V_v \cap \mathbb{P}_\psi$ (since $\mathbb{P}_{\varphi^+} \cap \mathbb{P}_\psi = \emptyset$, and $V_v = V_w \setminus P_w$). Hence, we deduce that $M, v \models \sim\varphi^+ \wedge \psi$, but $w\mathcal{S}v$, so we also have $M, w \models \langle \mathrm{S} \rangle(\sim\varphi^+ \wedge \psi)$. Finally, it is trivial to conclude that $M, w \models \langle \mathrm{T} \rangle \langle \mathrm{S} \rangle(\sim\varphi^+ \wedge \psi)$ since $\mathcal{T}(w) = \{w\}$.
Case (2): let $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$. So there exists u with $u \neq w$ and $w\mathcal{T}u$. Moreover, u is uniquely determined (see Remark 2.2). Then we choose u, but not w, as a candidate to satisfy the formula $\varphi^+ \wedge \psi$ (see $(*)$ above). Hence, we obtain from $(*)$ that $M, u \models \varphi^+ \wedge \psi$. The proof follows almost through the same reasoning as in the previous case, so we leave it to the reader. Following the same steps (above) for u here, we obtain $M, u \models \langle \mathrm{S} \rangle(\sim\varphi^+ \wedge \psi)$. Then $M, w \models \langle \mathrm{T} \rangle \langle \mathrm{S} \rangle(\sim\varphi^+ \wedge \psi)$ results automatically.

To prove completeness w.r.t. **MEM** models we use canonical models (Blackburn et al. [2001a]; Carnielli et al. [2009]). Let φ be a consistent $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$ formula. We define the canonical model $M^\varphi = (W, \mathcal{T}, \mathcal{S}, V)$ as follows. W is the set of maximal consistent sets of **MEM**. The accessibility relations $\mathcal{T}$ and $\mathcal{S}$ are such that:

$$\begin{array}{lll} w\mathcal{T}u & \text{iff} & \{\psi \ : \ [\mathrm{T}]\psi \in w\} \subseteq u \\ w\mathcal{S}u & \text{iff} & \{\psi \ : \ [\mathrm{S}]\psi \in w\} \subseteq u \end{array}$$

The valuation V is defined by $V_w = w \cap \mathbb{P}_\varphi$, for every $w \in W$. Let us prove that the canonical model $M^\varphi = (W, \mathcal{T}, \mathcal{S}, V)$ is a legal **MEM** model. As they are straightforward, we leave some parts of the proof to the reader, more explicitly, to prove that M^φ satisfies the constraints associated to the axioms $\mathrm{T}([\mathrm{T}])$, $4([\mathrm{T}])$, $\mathrm{T}_2([\mathrm{S}])$, and $\mathrm{WTriv}_2([\mathrm{S}])$.

- Axiom schema $\mathrm{Alt}_2([\mathrm{T}])$ assures that the constraint $\mathrm{alt}_2(\mathcal{T})$ holds in the canonical model described above: let $w\mathcal{T}u$ (1), $w\mathcal{T}u'$ (2), and $w\mathcal{T}u''$ (3). Then we assume for a contradiction that $u \neq u'$ (4), $u \neq u''$ (5), and $u' \neq u''$ (6). First, by (4), there is an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula φ such that $\varphi \in u'$ (7) and $\sim\varphi \in u$ (8). Second, (6) implies that there is an $\mathcal{L}_{[\mathrm{T}],[\mathrm{S}]}$-formula ψ such that $\psi \in u'$ (9) and $\sim\psi \in u''$. Finally, (5) guarantees the existence of $\chi \in u$ (10) with $\sim\chi \in u''$, but u'' is maximal consistent, therefore, so is $\sim(\psi \vee \chi) \in u''$. Moreover, from (3) we obtain $\langle \mathrm{T} \rangle \sim(\psi \vee \chi) \in w$ (11). As u is maximal consistent, and also using (10), we get $\psi \vee \chi \in u$ and then also $(\psi \vee \chi) \wedge \sim\varphi \in u$ (12)

by the help of (8). (1) and (12) gives us $\langle \mathrm{T}\rangle\Big((\psi\vee\chi)\wedge\sim\varphi\Big) \in w$ (13). Now we get $\langle \mathrm{T}\rangle\sim(\psi\vee\chi)\wedge\langle \mathrm{T}\rangle\Big((\psi\vee\chi)\wedge\sim\varphi\Big) \in w$ using maximal consistency of w, (11) and (13). Moreover, $\Big(\ \langle \mathrm{T}\rangle\sim(\psi\vee\chi)\wedge\langle \mathrm{T}\rangle\Big((\psi\vee\chi)\wedge\sim\varphi\Big)\Big) \supset [\mathrm{T}]\sim\Big((\psi\vee\chi)\wedge\varphi\Big)$, which is nothing but a variation of $\mathrm{Alt}_2([\mathrm{T}])$, is also in w since w is maximal consistent. Through the same reasoning, we also get $[\mathrm{T}]\sim\Big((\psi\vee\chi)\wedge\varphi\Big) \in w$ by modus ponens (MP), but then $\langle \mathrm{T}\rangle\Big((\psi\vee\chi)\wedge\varphi\Big)$ is not in w $(*)$ since w is maximal consistent. On the other hand, from (7), (9) and the maximal consistency of u' we obtain $(\psi\vee\chi)\wedge\varphi \in u'$. Furthermore, (2) gives us $\langle \mathrm{T}\rangle\Big((\psi\vee\chi)\wedge\varphi\Big) \in w$, but the latter contradicts $(*)$, so we are done.

- The weak mixed conversion axiom $\mathrm{WMConv}([\mathrm{T}],[\mathrm{S}])$ implies that the constraint $\mathrm{wmconv}(\mathcal{T},\mathcal{S})$ is satisfied in the canonical model: suppose that $w\mathcal{T}u$ and $w \neq u$; we want to show $u\mathcal{S}w$; assume for a contradiction that u isn't $\mathcal{S}$-related to w; then there exists φ such that $[\mathrm{S}]\varphi \in u$ and $\sim\varphi \in w$; next, since $w \neq u$, there exists ψ with $\psi \in w$ and $\sim\psi \in u$; as w is maximal consistent $\sim\varphi\wedge\psi \in w$, but so is any instance of $\mathrm{WMConv}([\mathrm{T}],[\mathrm{S}])$ as well; hence $\Big(\sim\varphi\wedge\psi\Big) \supset [\mathrm{T}]\Big((\sim\varphi\wedge\psi)\vee\langle \mathrm{S}\rangle(\sim\varphi\wedge\psi)\Big) \in w$, and then $(\sim\varphi\wedge\psi)\vee\langle \mathrm{S}\rangle(\sim\varphi\wedge\psi) \in u$ first through (MP) and then using our initial assumption $w\mathcal{T}u$; $\sim\psi \in u$ implies $\sim\psi\vee\varphi \in u$, but then so must $\langle \mathrm{S}\rangle(\sim\varphi\wedge\psi) \in u$; using maximal consistency of u we assert $\langle \mathrm{S}\rangle(\sim\varphi\wedge\psi) \supset (\langle \mathrm{S}\rangle\sim\varphi\wedge\langle \mathrm{S}\rangle\psi) \in u$ as well, but then so is $\langle \mathrm{S}\rangle\sim\varphi\wedge\langle \mathrm{S}\rangle\psi \in u$, which gives us the desired contradiction because $[\mathrm{S}]\varphi \in u$ implies $[\mathrm{S}]\varphi\vee[\mathrm{S}]\sim\psi \in u$ since u is maximal consistent; eventually $u\mathcal{S}w$.

- The mixed conversion axiom $\mathrm{MConv}([\mathrm{S}],[\mathrm{T}])$[1] guarantees that the constraint $\mathrm{mconv}(\mathcal{S},\mathcal{T})$ holds in the canonical model: let $w\mathcal{S}u$ and assume φ is such that $[\mathrm{T}]\varphi \in u$; then by the definition of $\mathcal{S}$, $\langle \mathrm{S}\rangle[\mathrm{T}]\varphi \in w$; since w is maximal consistent, any instance of $\mathrm{MConv}([\mathrm{S}],[\mathrm{T}])$ is in w, so is $\langle \mathrm{S}\rangle[\mathrm{T}]\varphi \supset \varphi$; therefore through (MP) we get $\varphi \in w$ and this completes the proof.

- The axiom schema $\mathrm{Heredity}([\mathrm{S}])$ ensures that the canonical model satisfies the constraint $\mathrm{heredity}(\mathcal{S})$, viz. that for every w,u, $w\mathcal{S}u$ implies $V_u \subseteq V_w$: indeed, suppose $w\mathcal{S}u$ and $p \in V_u = u\cap\mathbb{P}_\varphi$; as w is a maximal consistent set, it contains all instances of $\mathrm{Heredity}([\mathrm{S}])$, in particular, $\langle \mathrm{S}\rangle p \supset p$; since $w\mathcal{S}u$ we also obtain $\langle \mathrm{S}\rangle p \in w$ from $p \in u$. (Otherwise, w being maximal consistent, it includes $\sim\langle \mathrm{S}\rangle p = [\mathrm{S}]\sim p$; since $w\mathcal{S}u$ by assumption, we get

[1] It is handier to work with the contrapositive of the axiom schema $\mathrm{MConv}([\mathrm{S}],[\mathrm{T}])$ here, i.e., with $\langle \mathrm{S}\rangle[\mathrm{T}]\varphi \supset \varphi$.

$\sim p \in u$, contradicting the fact that u is consistent since $p \in u$ as well.) Hence, by (MP), $p \in w$, and so $p \in w \cap \mathbb{P}_\varphi = V_w$.

- The negatable axiom Neg([S], [T]) guarantees that neg($\mathcal{S}, \mathcal{T}$) holds in the canonical model: to see this take an arbitrary $w \in W$; since the canonical model satisfies the constraints refl($\mathcal{T}$) and alt$_2$($\mathcal{T}$) (the reader can easily check that M^φ satisfies refl($\mathcal{T}$) and as for alt$_2$($\mathcal{T}$) see above), we go through the following two cases:
  Case (i): let $\mathcal{T}(w) \setminus \{w\} = \emptyset$. Hence $\mathcal{T}(w) = \{w\}$ by the reflexivity of $\mathcal{T}$. (Then it is trivial to conclude that there exists u such that $w\mathcal{T}u$, and moreover $u = w$.) If $V_w = w \cap \mathbb{P}_\varphi = \emptyset$ (i.e., if w contains the negations of the propositional variables of φ) then the constraint trivially holds. Let $V_w \neq \emptyset$. Suppose $P \subseteq V_w = w \cap \mathbb{P}_\varphi$ is such that $P \neq \emptyset$. Then we choose $Q = V_w \setminus P$. Since $P, Q \subseteq \mathbb{P}$ are finite with $P \neq \emptyset$ and $P \cap Q = \emptyset$, now we can use Lemma 2.1. As w is a maximal consistent set it includes $(\bigwedge_{p \in P} p) \wedge (\bigwedge_{q \in Q} q)$, but then also $\langle \mathrm{T} \rangle \left((\bigwedge_{p \in P} p) \wedge (\bigwedge_{q \in Q} q) \right)$ since $\mathcal{T}(w) = \{w\}$. Next, again since w is maximal consistent, by Lemma 2.1 it also has every instance of Neg$'$([S], [T]), so it must contain $\langle \mathrm{T} \rangle \langle \mathrm{S} \rangle \left((\bigwedge_{p \in P} \sim p) \wedge (\bigwedge_{q \in Q} q) \right)$ as well. By our initial assumption, $\langle \mathrm{S} \rangle \left((\bigwedge_{p \in P} \sim p) \wedge (\bigwedge_{q \in Q} q) \right) \in w$. Thus we can conclude that there is $v \in W$ such that $w\mathcal{S}v$. Furthermore v contains $(\bigwedge_{p \in P} \sim p) \wedge (\bigwedge_{q \in Q} q)$. Therefore, $P \cap v = \emptyset$ and $Q \subseteq v$, but the canonical model satisfies the heredity($\mathcal{S}$) constraint (see above), so $V_v \subseteq V_w$. We know that $P, Q \subseteq \mathbb{P}_\varphi$ are mutually exclusive and cover V_w. Also, $Q \subseteq v$ and $Q \subseteq \mathbb{P}_\varphi$ implies $Q \subseteq v \cap \mathbb{P}_\varphi = V_v$. On the other hand, $V_v \cap P = (v \cap \mathbb{P}_\varphi) \cap P = v \cap (\mathbb{P}_\varphi \cap P) = v \cap P = \emptyset$. (Alternatively, note that $V_v = v \cap \mathbb{P}_\varphi = Q$, so apparently, $V_v \cap P = Q \cap P = \emptyset$.) It follows that $V_v = Q = V_w \setminus P$ and we are done.
  Case (ii): now we suppose $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$. Hence, $\mathcal{T}(w) = \{w, u\}$ for a uniquely defined u such that $u \neq w$ since the canonical model M^φ satisfies the constraints refl($\mathcal{T}$) and alt$_2$($\mathcal{T}$) (see above). Then it is obvious that there exists v such that $w\mathcal{T}v$. Choose $v = u$. Additionally, trans($\mathcal{T}$) also holds in the canonical model (as the reader can easily verify), so we further have $\mathcal{T}(u) = \{u\}$. Therefore the rest of the proof can basically be done in the same way as before.

To sum it up, the canonical model M^φ satisfies all constraints of the class of **MEM** models, so is indeed a legal **MEM** model. Moreover, as φ is a consistent **MEM** formula, there must exist a maximal **MEM** consistent set $w \in W$ containing φ. Then It can be proved in the standard way that $M^\varphi, w \models \varphi$. q.e.d.

Proposition A.5 *For $T \subseteq \mathbb{P}$, let $M_T = (W, \mathcal{T}, \mathcal{S}, V)$ be a Kripke model such that:*

$$\begin{array}{lcl} W & = & 2^T; \\ V_H & = & H, \quad \textit{for every } H \in W; \\ \mathcal{T} & = & \Delta_W \cup (W \times \{T\}) = \{(x,y) \in W \times W \ : \ x = y \textit{ or } y = T\}; \\ \mathcal{S} & = & \Delta_{(W \setminus \{T\})} \cup \Big(\{T\} \times (W \setminus \{T\})\Big) \ = \ \mathcal{T}^{-1} \setminus \{(T,T)\}. \end{array}$$

Then M_T is a **MEM** *model. Furthermore, for every $\mathcal{L}_{\rightarrow}$-formula φ and $H \subseteq T$,*

$$H, T \models \varphi \textit{ if and only if } M_T, H \models \mathsf{tr}(\varphi).$$

Proof. First, M_T is a legal **MEM** model: M_T satisfies all constraints, i.e., refl($\mathcal{T}$), alt$_2$($\mathcal{T}$), trans($\mathcal{T}$), refl$_2$($\mathcal{S}$), wtriv$_2$($\mathcal{S}$), wmconv($\mathcal{T}, \mathcal{S}$), mconv ($\mathcal{S}, \mathcal{T}$), heredity($\mathcal{S}$), and neg($\mathcal{S}, \mathcal{T}$) by construction. Second, one can prove by a straightforward induction on the form of φ that $H, T \models \varphi$ iff $M_T, H \models \mathsf{tr}(\varphi)$, for every $H \subseteq T$. q.e.d.

Proposition A.6 *Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a* **MEM** *model. Then for every $w \in W$ and every $\mathcal{L}_{\rightarrow}$-formula φ we have:*

1. *If $\mathcal{T}(w) \setminus \{w\} = \emptyset$ then $M, w \models \mathsf{tr}(\varphi)$ if and only if $V_w, V_w \models \varphi$;*
2. *If $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ then $M, w \models \mathsf{tr}(\varphi)$ if and only if $V_w, V_u \models \varphi$ for the uniquely determined $u \in \mathcal{T}(w) \setminus \{w\}$.*

Proof. As expected we give the proof by induction on φ through the case analysis. Let $w \in W$ then by Remark 2.2 we have two cases:
Case 1: let $\mathcal{T}(w) = \{w\}$ $(*)$ then $\mathcal{T}(w) \setminus \{w\} = \emptyset$, i.e., w is the root point of a tree structure (see Footnote 2.1.3). Now, to be able to give the result, we use induction on the form of φ. The base cases are immediate from the definition of truth conditions. The first two Boolean cases easily follow from the induction hypothesis (see the first item above), and the intuitionistic implication step is also obtained from the induction hypothesis (see the first item above) using $(*)$.
Case 2: as to the second part, let $\mathcal{T}(w) = \{w, u\}$ $(**)$. Hence, $\mathcal{T}(w) \setminus \{w\} = \{u\}$, i.e., in words, $\mathcal{T}(w) \setminus \{w\}$ contains exactly one element, u which is the root of a tree structure containing w (see Footnote 2.1.3), and as a consequence $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$. Then by Remark 2.2, we also have $\mathcal{T}(u) = \{u\}$ $(\star)$. Moreover, since $w \mathcal{T} u$ from Proposition 2.1.1, we obtain $V_w \subseteq V_u$ $(\star\star)$. Now, we verify the claim again by induction on φ. The base and the first two Boolean cases are still easy, and basically follow the same way as above. However, the case of intuitionistic implication is worth analyzing. We here sketch out the argument and leave the gaps to the reader. Nevertheless, one should note that in this step, our induction

hypothesis is made up of both items above. To spell it out, we use item 2 for u while we use item 1 for w as our induction hypotheses. So, we have:

$$\begin{array}{lll} M, w \models \mathsf{tr}(\psi_1 \to \psi_2) & \text{iff} & M, w \models [\mathrm{T}]\big(\mathsf{tr}(\psi_1) \supset \mathsf{tr}(\psi_2)\big) \\ & \text{iff} & M, w \models \mathsf{tr}(\psi_1) \supset \mathsf{tr}(\psi_2) \ \text{ and } \ M, u \models \mathsf{tr}(\psi_1) \supset \mathsf{tr}(\psi_2) \\ & & \qquad\qquad\qquad\qquad\qquad\qquad (\text{by } (**)) \\ & \text{iff} & V_w, V_u \models \psi_1 \supset \psi_2 \ \text{ and } \ V_u, V_u \models \psi_1 \supset \psi_2 \\ & & \qquad\qquad\qquad\qquad\qquad\qquad (\text{by I.H., } (\star) \text{ and } (\star\star)) \\ & \text{iff} & V_w, V_u \models \psi_1 \to \psi_2. \end{array}$$

q.e.d.

Theorem A.2 *Given an $\mathcal{L}_{\to}$-formula, φ,*

φ is HT valid if and only if $\mathsf{tr}(\varphi)$ is **MEM** *valid.*

Proof. It follows from Proposition 2.5 and Proposition 2.6 in the way given below:
($\Longleftarrow$):
Let (H, T) be an HT model, then $H \subseteq T \subseteq \mathbb{P}$. Now, construct a Kripke model $M_T = (W, \mathcal{T}, \mathcal{S}, V)$ as in Proposition 2.5. By that proposition, M_T is a **MEM** model, and since $tr(\varphi)$ is **MEM** valid by assumption, we have $M_T, H \models tr(\varphi)$. Finally, again by Proposition 2.5, $H, T \models \varphi$, i.e., (H, T) is an HT model of φ.
($\Longrightarrow$):
Let $M = (W, \mathcal{T}, \mathcal{S}, V)$ be a **MEM** model and let $w \in W$. We give the proof through a case study.
Case 1. Assume that $\mathcal{T}(w) \setminus \{w\} = \emptyset$. By assumption, we know that (V_w, V_w) is an HT model of φ. Therefore, by Proposition 2.6 we have $M, w \models tr(\varphi)$.
Case 2. Suppose that $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$. Then there exists a unique u such that $\mathcal{T}(w) = \{w, u\}$ is of cardinality 2 (see Remark 2.2). Hence, Proposition 2.1.1 gives us $V_w \subseteq V_u$. Next, by hypothesis (V_w, V_u) is an HT model of φ. Therefore, by Proposition 2.6, $M, w \models tr(\varphi)$. q.e.d.

Corollary A.1 *For every $\mathcal{L}_{\to}$-formula φ,*

φ has an HT model if and only if $\mathsf{tr}(\varphi)$ is **MEM** *satisfiable.*

PROOF. It is a direct consequence of Theorem 2.2, and the result follows through the argument given below:

$\mathsf{tr}(\varphi)$ is **MEM** satisfiable
iff $\sim\mathsf{tr}(\varphi)$ is not **MEM** valid
iff $\sim\langle \mathrm{T}\rangle\mathsf{tr}(\varphi)$ is not **MEM** valid (by refl($\mathcal{T}$))
iff $\mathsf{tr}(\neg\varphi)$ is not **MEM** valid (see Subsection 3.3.3 for $\mathsf{tr}(\neg\varphi)$)
iff $\neg\varphi$ is not HT valid (by Theorem 2.2)
iff there is an HT model (H,T) s.t. $H,T \not\models \neg\varphi$
iff there is an HT model (H,T) s.t. $H,T \models \varphi$ or $T,T \models \varphi$
iff φ has an HT model.

q.e.d.

Proposition A.7 *Given $T \subseteq \mathbb{P}$, let $M_T = (W, \mathcal{T}, \mathcal{S}, V)$ be a Kripke model such that:*

$$\begin{array}{lcl} W & = & 2^T; \\ V_H & = & H, \quad \textit{for every } H \in W; \\ \mathcal{T} & = & \Delta_W \cup (W \times \{T\}); \\ \mathcal{S} & = & \Delta_{(W\setminus\{T\})} \cup \Big(\{T\} \times (W \setminus \{T\})\Big). \end{array}$$

Then M_T is a **MEM** *model, and T is an equilibrium model of φ if and only if $M_T, T \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$, for every $\mathcal{L}_{\rightarrow}$-formula φ.*

PROOF. As we have already checked in Proposition 2.5, M_T is a legal **MEM** model. So it remains to prove that T is an equilibrium model of φ iff $M_T, T \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$ for every $\mathcal{L}_{\rightarrow}$-formula φ. We indeed have:

T is an equilibrium model of φ
iff $T,T \models \varphi$ and $H,T \not\models \varphi$ for every $H \subset T$
iff $M_T, T \models \mathsf{tr}(\varphi)$ and $M_T, H \not\models \mathsf{tr}(\varphi)$ for every $H \subset T$ (by Proposition 2.5)
iff $M_T, T \models \mathsf{tr}(\varphi)$ and $M_T, H \models {\sim}\mathsf{tr}(\varphi)$ for every H such that $T\mathcal{S}H$
(because $T\mathcal{S}H$ iff $H \subset T$ by construction)
iff $M_T, T \models \mathsf{tr}(\varphi)$ and $M_T, T \models [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$
iff $M_T, T \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$.

q.e.d.

Proposition A.8 *Given a* **MEM** *model $M = (W, \mathcal{T}, \mathcal{S}, V)$ and $w \in W$, if $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ then let $u \in \mathcal{T}(w) \setminus \{w\}$, or else let $u = w$. Then*

1. *if* $V_u = \emptyset$ *then* $M, u \models \mathsf{tr}(\varphi)$ *if and only if* V_u *is an equilibrium model for* φ*,*

2. *and if* $V_u \neq \emptyset$ *then* $M, u \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi)$ *if and only if* V_u *is an equilibrium model for* φ*, for every* $\mathcal{L}_{\rightarrow}$*-formula* φ*.*

Proof. Let $M = (M, \mathcal{T}, \mathcal{S}, V)$ be a **MEM** model, and $w \in W$.
Case 1: let $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$ then by Remark 2.2, there exists a uniquely determined $u \in \mathcal{T}(w) \setminus \{w\}$ such that $\mathcal{T}(w) = \{w, u\}$, and $\mathcal{T}(u) = \{u\}$ $(\star)$. Moreover, for every v such that $u\mathcal{S}v$, by $\mathrm{wmcon}(\mathcal{T}, \mathcal{S})$, $\mathrm{refl}_2(\mathcal{S})$, $\mathrm{wtriv}_2(\mathcal{S})$, $\mathrm{mconv}(\mathcal{S}, \mathcal{T})$, $\mathrm{alt}_2(\mathcal{T})$, we have $\mathcal{T}(v) = \{v, u\}$ $(\star\star)$. Hence, the result follows through the following argument:

- let $V_u \neq \emptyset$ $(*)$ then we have:

$$\begin{array}{ll} & M, u \models \mathsf{tr}(\varphi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\varphi) \\ \text{iff} & M, u \models \mathsf{tr}(\varphi) \text{ and } M, v \not\models \mathsf{tr}(\varphi) \text{ for every } v \text{ such that } u\mathcal{S}v \\ \text{iff} & V_u, V_u \models \varphi \text{ and } V_v, V_u \not\models \mathsf{tr}(\varphi) \text{ for every } v \text{ such that } u\mathcal{S}v \\ & \qquad \text{(by Proposition 2.6, } (\star) \text{ and } (\star\star)) \\ \text{iff} & V_u, V_u \models \varphi \text{ and } H, V_u \not\models \mathsf{tr}(\varphi) \text{ for every } H \text{ such that } H \subset V_u \\ & \qquad \text{(by } (*), \text{ negatable}(\mathcal{S}, \mathcal{T}) \text{ and } \mathrm{wtriv}_2(\mathcal{S})) \\ \text{iff} & V_u \text{ is an equilibrium model for } \varphi. \end{array}$$

- let $V_u = \emptyset$ then we have:

$$\begin{array}{lll} M, u \models \mathsf{tr}(\varphi) & \text{iff} & V_u, V_u \models \varphi \text{ (by Proposition 2.6 and } (\star)) \\ & \text{iff} & V_u \text{ is an equilibrium model for } \varphi. \end{array}$$

Case 2: let $\mathcal{T}(w) \setminus \{w\} = \emptyset$ then by Remark 2.2, $\mathcal{T}(w) = \{w\}$ Moreover, for every v such that $w\mathcal{S}v$, $\mathcal{T}(v) = \{v, w\}$ (note that if w is a singleton point in which $\mathcal{S}(w) \neq \emptyset$ then $\mathcal{T}(v) = \{w\}$ for every v with $w\mathcal{S}v$ because for that case $\mathcal{S}(w) = \{w\}$, so $v = w$). The rest of the proof follows basically the same way, so we leave it to the reader. q.e.d.

Theorem A.3 *Given* $\mathcal{L}_{\rightarrow}$*-formulas* χ *and* φ*,*

$$\chi \mid\!\approx \varphi \text{ if and only if } \Big(\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)\Big) \supset \mathsf{tr}(\varphi) \text{ is } \textbf{MEM} \text{ valid.}$$

Proof. We use propositions 2.6, 2.7 and 2.8. First of all, let us abbreviate $\Big(\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)\Big) \supset \mathsf{tr}(\varphi)$ by ξ.
$(\Longrightarrow)$:
Assume ξ isn't **MEM** valid. Hence there exists a **MEM** model M and a world w in M such that:

$$M, w \models \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi) \wedge {\sim}\mathsf{tr}(\varphi) \quad (*).$$

If w were such a point satisfying $\mathcal{T}(w) \setminus \{w\} \neq \emptyset$, then from $(*)$ we would immediately obtain a contradiction, namely that $M, w \models \mathsf{tr}(\chi)$ and also $M, w \not\models \mathsf{tr}(\chi)$ (by Remark 2.2 and the constraints $\mathrm{wmconv}(\mathcal{T}, \mathcal{S})$ and $\mathrm{refl}_2(\mathcal{S})$). Hence we conclude that $\mathcal{T}(w) \setminus \{w\} = \emptyset$ $(\star)$, i.e., by Remark 2.2, $\mathcal{T}(w) = \{w\}$. Now, we need to go over two cases:
Case 1: let V_w be empty then we again have two alternatives. One is where w is a singleton point. However, w cannot be a singleton point where $\mathcal{S}(w)$ is nonempty otherwise we would again get the same contradiction as before. Therefore, if w is a singleton point then it should satisfy $\mathcal{S}(w) = \emptyset$. Now, using $(*)$, it turns out that $M, w \models \mathsf{tr}(\chi)$ and $M, w \not\models \mathsf{tr}(\varphi)$. Then by Proposition 2.6 and $(\star)$, we get $V_w, V_w \models \chi$ and $V_w, V_w \not\models \varphi$. We know that for the empty-set, the minimality condition in the definition of equilibrium model is trivially satisfied, and '$V_w, V_w \models \chi$' guarantees the first condition. Therefore we conclude that V_w is an equilibrium model for χ. However, $V_w, V_w \not\models \varphi$ helps us deduce that φ is not a logical consequence of χ, which is nothing but the result we are looking for. The second instance is where w is not a singleton point. In this part of the proof, we first use $(*)$ and get $M, w \models [\mathrm{S}]{\sim}\mathsf{tr}(\chi)$ which further gives us that for every u such that $w\mathcal{S}u$, $M, u \not\models \mathsf{tr}(\chi)$. Next, using the constraints $\mathrm{refl}_2(\mathcal{S})$, $\mathrm{mconv}(\mathcal{S})$, $\mathrm{alt}_2(\mathcal{T})$, $\mathrm{wtriv}_2(\mathcal{S})$, from w being not a singleton point with $\mathcal{T}(w) = \{w\}$ we obtain $\mathcal{T}(u) = \{u, w\}$ for every u such that $w\mathcal{S}u$ (one should note that $w\mathcal{S}w$ is not possible when w is the root of a tree, but is not a singleton point). Then since $\mathcal{T}(u) \setminus \{u\} \neq \emptyset$ for every u such that $w\mathcal{S}u$, by Proposition 2.6, we have $V_u, V_w \not\models \chi$. On the other hand, by heredity($\mathcal{S}$), we get $V_u \subseteq V_w$ for every u such that $w\mathcal{S}u$. Thus, $V_u = \emptyset$ for every u with $w\mathcal{S}u$. Hence, '$V_w, V_w \models \chi$' and '$V_u, V_w \not\models \chi$, for every u such that $w\mathcal{S}u$' contradict each other. As a result, using the discussion above, we conclude that if $V_w = \emptyset$ then $(*)$ gives us that w is a singleton point in which $\mathcal{S}(w) = \emptyset$ and for that case we easily get the result we are searching for (see above).
Case 2: let V_w be different from empty-set. First of all, we recall $M, w \models \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)$ $(**)$, which is an immediate consequence of $(*)$. Then, Proposition 2.8 and $(**)$ immediately yield us that V_w is an equilibrium model of χ. However, we also know that $V_w, V_w \not\models \varphi$ (see above). Thus, we conclude that $\chi \not\mid\!\approx \varphi$. So, we are done.
$(\Longleftarrow)$:
Let ξ be **MEM** valid, and let $T \subseteq \mathbb{P}$ be an equilibrium model of χ. Now we construct the **MEM** model, M_T, as it is done in Proposition 2.7. Then we conclude that M_T is a legal **MEM** model and that $M_T, T \models \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)$ again by Proposition 2.7. Since ξ is **MEM** valid, $M_T, T \models \xi$, but as M_T is a **MEM** model, we also have $M_T, T \models \mathsf{tr}(\varphi)$. Finally, since $\mathcal{T}(T) = \{T\}$ by construction, then $T, T \models \varphi$ immediately follows by Proposition 2.6. Thus we conclude that $\chi \not\mid\!\approx \varphi$, and this ends the proof. q.e.d.

Corollary A.2 *For every $\mathcal{L}_{\rightarrow}$-formula χ,*

$$\chi \textit{ has an equilibrium model } \textit{iff } \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi) \textit{ is } \mathbf{MEM} \textit{ satisfiable.}$$

Proof. It is an immediate consequence of Theorem 3.4 as it is shown below:

χ has an equilibrium model, say T
iff T is an equilibrium model of χ and $T, T \not\models \bot$
iff $\chi \not\sim \bot$
iff $\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi) \supset \mathsf{tr}(\bot)$ is not **MEM** valid (by Theorem 3.4)
iff $M, w \models \mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi) \wedge \top$ for some **MEM** model M and a world w
iff $\mathsf{tr}(\chi) \wedge [\mathrm{S}]{\sim}\mathsf{tr}(\chi)$ is **MEM** satisfiable.

q.e.d.

A.3 Proofs of Chapter 3

Proposition A.9 *Given an $\mathcal{L}_{\text{D-HT}}$-formula φ, let P be a set of propositional variables such that $P \cap \mathbb{P}_{\varphi} = \emptyset$. Then, for every $Q \subseteq P$,*

$$(H, T) \in \|\varphi\|_{\text{D-HT}} \quad \textit{if and only if} \quad (H \cup Q, T \cup P) \in \|\varphi\|_{\text{D-HT}}.$$

Proof. As expected, the proof is given by strong induction on the length of **D-HT** expressions (**D-HT** formulas and **D-HT** programs). Let $\Sigma(\zeta)$ be the property $(\star)$ defined over all **D-HT** expressions as in: given $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_{\varphi} = \emptyset$, for every $Q \subseteq P$,

- if ζ is a formula then $(H, T) \in \|\zeta\|_{\text{D-HT}}$ iff $(H \cup Q, T \cup P) \in \|\zeta\|_{\text{D-HT}}$, and
- if ζ is a program then $\Big((H_1, T_1), (H_2, T_2)\Big) \in \|\zeta\|_{\text{D-HT}}$ iff $\Big((H_1 \cup Q, T_1 \cup P), (H_2 \cup Q, T_2 \cup P)\Big) \in \|\zeta\|_{\text{D-HT}}$.

Now, we are going to show by induction that $\Sigma(\zeta)$ is true for all **D-HT** expressions. Base case: here, we will verify the claim for all **D-HT** expressions of length 1.

- Let $\zeta = p$ for an arbitrary $p \in \mathbb{P}$. Then given $P \subseteq \mathbb{P}$ such that $p \notin P$, for every $Q \subseteq P$ $\Big($so, $p \notin Q$ either $(\triangle)\Big)$, we have $(H, T) \in \|p\|_{\text{D-HT}}$ if and only if $p \in H$ (by $(\triangle)$) if and only if $p \in H \cup Q$ if and only if $(H \cup Q, T \cup P) \in \|p\|_{\text{D-HT}}$ (see Table 3.1 for the interpretation of $p \in \mathbb{P}$).

- for $\zeta = \bot$, the result immediately follows through the semantics of $\bot$ (see Table 3.1).

- Let $\zeta = +p$. Then, given $P \subseteq \mathbb{P}$ such that $p \notin P$, we assume $\Big((H_1 \cup Q, T_1 \cup P), (H_2 \cup Q, T_2 \cup P)\Big) \in \|{+p}\|_{\textbf{DL-PA}}$, for every $Q \subseteq P$ (since $p \notin P$, $p \notin Q$ either (∇)). Taking $P = Q = \emptyset$ immediately gives the result, i.e., $\Big((H_1, T_1), (H_2, T_2)\Big) \in \|{+p}\|_{\textbf{D-HT}}$. For the reverse direction, we first suppose $\Big((H_1, T_1), (H_2, T_2)\Big) \in \|{+p}\|_{\textbf{D-HT}}$. Then, by the help Table 3.1, we get the following: "if $p \notin T_1$ then $T_2 = T_1 \cup \{p\}$ and $H_2 = H_1$", and "if $p \in T_1$, but $p \notin H_1$ then $H_2 = H_1 \cup \{p\}$ and $T_2 = T_1$". Moreover, using (∇) we also get: "if $p \notin T_1 \cup P$ then $T_2 \cup P = (T_1 \cup P) \cup \{p\}$ and $H_2 \cup Q = H_1 \cup Q$", and "if $p \in T_1 \cup P$, but $p \notin H_1 \cup Q$ then $H_2 \cup Q = (H_1 \cup Q) \cup \{p\}$ and $T_2 \cup P = T_1 \cup P$". Hence, Table 3.1 gives us $\Big((H_1 \cup Q, T_1 \cup P), (H_2 \cup Q, T_2 \cup P)\Big) \in \|{+p}\|_{\textbf{DL-PA}}$.

- For $\zeta = -p$, the proof follows basically the same as above , so we leave it to the reader.

Inductive step: we assume that $\Sigma(\zeta)$ holds for every **D-HT** expression, ζ, of length l such that $1 \leq l < n$ (induction hypothesis **(I.H.)**). We want to prove that the property $(\star)$ holds for all **D-HT** expressions of length n.
Case 1: let ζ be a formula of length n.

- For $\zeta = \psi_1 \wedge \psi_2$ and $\zeta = \psi_1 \vee \psi_2$ where $\psi_1, \psi_2 \in \mathcal{L}_{\textbf{D-HT}}$, the proof is straightforward, so we leave it to the reader.

- However, the instance of (intuitionistic) implication is worth analyzing. Let $\zeta = \psi_1 \to \psi_2$ for some $\mathcal{L}_{\textbf{D-HT}}$ formulas ψ_1 and ψ_2. To show one side of the equivalence, we first assume $(H, T) \in \|\zeta\|_{\textbf{D-HT}}$, then from Table 3.1, we immediately obtain $(H, T), (T, T) \in (\mathbb{HT} \setminus \|\psi_1\|_{\textbf{D-HT}}) \cup \|\psi_2\|_{\textbf{D-HT}}$. We next use induction hypothesis and get the following: given $P_1 \subseteq \mathbb{P}$ such that $P_1 \cap \mathbb{P}_{\psi_1} = \emptyset$, for any $Q_1 \subseteq P_1$, $(H \cup Q_1, T \cup P_1)$ and $(T \cup Q_1, T \cup P_1)$ are not contained in $\|\psi_1\|_{\textbf{D-HT}}$ (hence, they are in $\mathbb{HT} \setminus \|\psi_1\|_{\textbf{D-HT}}$). Similarly, given $P_2 \subseteq \mathbb{P}$ such that $P_2 \cap \mathbb{P}_{\psi_2} = \emptyset$, for every $Q_2 \subseteq P_2$, $(H \cup Q_2, T \cup P_2)$ and $(T \cup Q_2, T \cup P_2)$ are both included in $\|\psi_2\|_{\textbf{D-HT}}$. Hence, we deduce that for every $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\zeta = \emptyset$, and every $Q \subseteq P$, $(H \cup Q, T \cup P)$ $(\bullet)$ and $(T \cup Q, T \cup P)$ are both in $(\mathbb{HT} \setminus \|\psi_1\|_{\textbf{D-HT}}) \cup \|\psi_2\|_{\textbf{D-HT}}$. However, among the latter HT models, we just consider the total (in which here and there are equal to each other) models. Such restriction gives us the following: for every P such that $P \cap \mathbb{P}_\zeta = \emptyset$, $(T \cup P, T \cup P) \in (\mathbb{HT} \setminus \|\psi_1\|_{\textbf{D-HT}}) \cup \|\psi_2\|_{\textbf{D-HT}}$ $(\bullet\bullet)$. As a result, what we obtain

from (•) and (••) is the following: for every $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\zeta = \emptyset$ and every $Q \subseteq P$, $(H \cup Q, T \cup P), (T \cup P, T \cup P) \in (\mathbb{HT} \setminus \|\psi_1\|_{\textbf{D-HT}}) \cup \|\psi_2\|_{\textbf{D-HT}}$. Finally, using Table 3.1, we get $(H \cup Q, T \cup P) \in \|\psi_1 \to \psi_2\|_{\textbf{D-HT}}$. The other side is trivial. Setting $P = Q = \emptyset$, immediately gives the result (as the reader can easily check).

- As to the formulas preceded by dynamic modal operators, we just give the proof for $[\pi]$, and leave the rest to the reader; yet the proof of $\langle\pi\rangle$ is given basically the same as below. Let $\zeta = [\pi]\psi$ for some $\pi, \psi \in \mathcal{L}_{\textbf{D-HT}}$. Then we have: $(H, T) \in \|[\pi]\psi\|_{\textbf{D-HT}}$ if and only if for every $\Big((H, T), (H_1, T_1)\Big) \in \|\pi\|_{\textbf{D-HT}}$, $(H_1, T_1) \in \|\varphi\|_{\textbf{D-HT}}$ (see Table 3.1 for semantics) if and only if (by (I.H.)) for every P such that $P \cap \mathbb{P}_\zeta = \emptyset$, and every $Q \subseteq P$, for every $(H_1 \cup Q, T_1 \cup P)$ with $\Big((H \cup Q, T \cup P), (H_1 \cup Q, T_1 \cup P)\Big) \in \|\pi\|_{\textbf{D-HT}}$, we have $(H_1 \cup Q, T_1 \cup P) \in \|\psi\|_{\textbf{D-HT}}$ if and only if for every P such that $P \cap \mathbb{P}_\zeta = \emptyset$, and every $Q \subseteq P$, $(H \cup Q, T \cup P) \in \|[\pi]\psi\|_{\textbf{D-HT}}$.

Case 2: let ζ be a program of length n.

- Both sides of the proof for $\zeta = \pi_1; \pi_2$ are easy; one side is even trivial (just take $P = Q = \emptyset$), and the other side is given simply using Table 3.1 and (I.H.). As for $\zeta = \pi_1 \cup \pi_2$, its proof follows basically the same as the proof of $\pi_1; \pi_2$. So, we leave them all to the reader.

- Now, let $\zeta = \pi^*$ for some $\mathcal{L}_{\textbf{D-HT}}$ program π. We just give one side of the proof because the other side is trivial (exactly the same as in item 1). So, we start by assuming that $\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\pi^*\|_{\textbf{D-HT}}$[1]. Then using Table 3.1 and Footnote 1, we get $\|\pi^*\|_{\textbf{D-HT}} = \|\pi\|^*_{\textbf{D-HT}} = \|\pi\|^+_{\textbf{D-HT}} \cup \iota$ (•) where ι is the identity relation. Hence, we have: '$H_i = H_j$ and $T_i = T_j$' or 'there exists a sequence $(H_i, T_i) = (H_0, T_0), (H_1, T_1), \ldots, (H_n, T_n) = (H_j, T_j)$ (for some $n > 0$) of **D-HT** models such that for each k, $0 \leq k < n$, $\Big((H_k, T_k), (H_{k+1}, T_{k+1})\Big) \in \|\pi\|_{\textbf{D-HT}}$'. Then using (I.H.), the latter implies that for every $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\pi = \emptyset$, and every $Q \subseteq P$, there is a sequence of **D-HT** models, namely $(H_i \cup Q, T_i \cup P) = (H_0 \cup Q, T_0 \cup P), \ldots, (H_n \cup Q, T_n \cup P) = (H_j \cup Q, T_j \cup P)$ $(n > 0)$, satisfying $\Big((H_k \cup Q, T_k \cup P), (H_{k+1} \cup Q, T_{k+1} \cup P)\Big) \in \|\pi\|_{\textbf{D-HT}}$

[1] We abbreviate the n-fold composition of a binary relation R by R^n. Formally, $R^0 \stackrel{\text{def}}{=} \{(x, x) : x \in \text{dom}(R)\}$. The transitive and the reflexive transitive closures of R are respectively defined by $R^* \stackrel{\text{def}}{=} \bigcup_{n \geq 0} R^n$ and $R^+ \stackrel{\text{def}}{=} \bigcup_{n > 0} R^n$. Hence, $R^* = R^+ \cup \iota$ where ι is the identity relation. Moreover, note that R^+uv means that there is a sequence of elements $u = w_0, w_1, \ldots, w_n = v$ $(n > 0)$ such that for each $i < n$ we have $(w_i, w_{i+1}) \in R$.

for all k, $0 \leq k < n$. Moreover, it is obvious that for every P such that $P \cap \mathbb{P}_\pi = \emptyset$, and every $Q \subseteq P$, $H_i \cup Q = H_j \cup Q$ and $T_i \cup Q = T_j \cup P$. Finally, we get for every P such that $P \cap \mathbb{P}_\pi = \emptyset$, and every $Q \subseteq P$, $\Big((H_i \cup Q, T_i \cup P), (H_j \cup Q, T_j \cup P)\Big) \in \|\pi\|^+_{\textbf{D-HT}} \cup \iota$ where ι is the identity relation. By $(\bullet)$, we also know that $\|\pi\|^+_{\textbf{D-HT}} \cup \iota = \|\pi^*\|_{\textbf{D-HT}}$, so we are done.

- Finally, let $\zeta = \psi?$ for some **D-HT** formula ψ. One part of the proof is again trivial (exactly the same as all above). To show the other part, we first assume $\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\psi?\|_{\textbf{D-HT}}$, and this means that $H_i = H_j$, $T_i = T_j$ and $(H_i, T_i) \in \|\psi\|_{\textbf{D-HT}}$ (see Table 3.1). Then, by (I.H.) and the first two equality, we get for every $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\psi = \emptyset$, and every $Q \subseteq P$, $H_i \cup Q = H_j \cup Q$, $T_i \cup P = T_j \cup P$ and $(H_i \cup Q, T_i \cup P) \in \|\psi\|_{\textbf{D-HT}}$. Using Table 3.1 once again (and also regarding that $\mathbb{P}_\zeta = \mathbb{P}_\psi$) we get the result, i.e., $\Big((H_i \cup Q, T_i \cup P), (H_j \cup Q, T_j \cup P)\Big) \in \|\zeta\|_{\textbf{D-HT}}$ for every $P \subseteq \mathbb{P}$ such that $P \cap \mathbb{P}_\zeta = \emptyset$, and every $Q \subseteq P$.

q.e.d.

Lemma A.3 (Main Lemma) $(H, T) \in \|\varphi\|_{\textbf{D-HT}}$ *iff* $H \cup T' \in \|\mathsf{tr}(\varphi)\|_{\textbf{DL-PA}}$.

Proof. Proof is given by strong induction on the length of **D-HT** expressions (**D-HT** formulas and **D-HT** programs). Let $\Pi(\xi)$ be the property $(\star)$ defined over all **D-HT** expressions as follows:

- if ξ is a formula then $(H, T) \in \|\xi\|_{\textbf{D-HT}}$ if and only if $H \cup T' \in \|\mathsf{tr}(\xi)\|_{\textbf{DL-PA}}$, and
- if ξ is a program then

  $((H_1, T_1), (H_2, T_2)) \in \|\xi\|_{\textbf{D-HT}}$ if and only if $((H_1 \cup T'_1), (H_2 \cup T'_2)) \in \|\mathsf{tr}(\xi)\|_{\textbf{DL-PA}}$.

We are going to demonstrate by induction that $\Pi(\xi)$ holds for all **D-HT** expressions, ξ.
Base case: in this step, we need to prove that the property Π holds for all **D-HT** expressions of length 1. To begin with, for all $p \in \mathbb{P}$, $\Pi(p)$ is clearly satisfied: for every $p \in \mathbb{P}$,

$$\begin{aligned}(H, T) \in \|p\|_{\textbf{D-HT}} \text{ iff } & \quad p \in H \quad \text{(see Table 3.1)} \\ \text{iff } & \quad p \in H \cup T' \\ \text{iff } & \quad H \cup T' \in \|\mathsf{tr}(p)\|_{\textbf{DL-PA}} \quad \text{(see Table 3.2 and Table 3.4)} .\end{aligned}$$

It is also obvious that $\Pi(\bot)$ trivially holds. As for the atomic programs, we just give the proof for $+p$ because the proof for $-p$ is similar:

$\Big((H_1, T_1), (H_2, T_2)\Big) \in \|{+p}\|_{\textbf{D-HT}}$ iff
(see Table 3.1)
"if $p \notin T_1$ then $T_2 = T_1 \cup \{p\}$ and $H_2 = H_1$", and
"if $p \in T_1$, but $p \notin H_1$ then $H_2 = H_1 \cup \{p\}$ and $T_2 = T_1$" iff
(see Remark 3.1)
"if $p' \notin T_1'$ then $T_2' = T_1' \cup \{p'\}$ and $H_2 = H_1$", and
"if $p' \in T_1'$, but $p \notin H_1$ then $H_2 = H_1 \cup \{p\}$ and $T_2' = T_1'$" iff
(Prop. 3.2 and Table 3.2)
"if $H_1 \cup T_1' \in \|{\sim p'}\|_{\textbf{DL-PA}}$ then $H_2 \cup T_2' = H_1 \cup (T_1' \cup \{p'\})$", and
"if $H_1 \cup T_1' \in \|{p' \wedge \sim p}\|_{\textbf{DL-PA}}$ then $H_2 \cup T_2 = (H_1 \cup \{p\}) \cup T_2'$" iff
(see Table 3.2)
$(H_1 \cup T_1', H_2 \cup T_2') \in \|\Big(\sim p'?; p' {:=} \top\Big) \cup \Big((p' \wedge \sim p)?; p {:=} \top\Big)\|_{\textbf{DL-PA}}$ iff
(see Table 3.4)
$(H_1 \cup T_1', H_2 \cup T_2') \in \|\mathsf{tr}(+p)\|_{\textbf{DL-PA}}$.

As a result, $\Pi(+p)$ holds as well.
Inductive step: we suppose that $\Pi(\xi)$ is true for every **D-HT** expression, ξ of length l such that $1 \leq l < n$ (induction hypothesis **(I.H.)**). We want to prove that the property $(\star)$ holds for all **D-HT** expressions of length n.
Case 1: let ξ be a formula of length n.
Let $\xi = \alpha \wedge \beta$ for some **D-HT** formulas α and β then clearly, $|\alpha|, |\beta| < n$. Hence, (I.H.) gives us "$(H, T) \in \|\theta\|_{\textbf{D-HT}}$ if and only if $H \cup T' \in \|\mathsf{tr}(\theta)\|_{\textbf{DL-PA}}$" for $\theta = \alpha, \beta$ $(\star\star)$. Then, the result follows as in:

$$
\begin{array}{lll}
(H, T) \in \|\xi\|_{\textbf{D-HT}} & \text{iff} & (H, T) \in \|\alpha\|_{\textbf{D-HT}} \cap \|\beta\|_{\textbf{D-HT}} \quad \text{(see Table 3.1)} \\
& \text{iff} & H \cup T' \in \|\mathsf{tr}(\alpha)\|_{\textbf{DL-PA}} \cap \|\mathsf{tr}(\beta)\|_{\textbf{DL-PA}} \quad \text{(by } \star\star\text{)} \\
& \text{iff} & H \cup T' \in \|\mathsf{tr}(\alpha) \wedge \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} \\
& \text{iff} & H \cup T' \in \|\mathsf{tr}(\xi)\|_{\textbf{DL-PA}}.
\end{array}
$$

For $\xi = \alpha \vee \beta$ such that $\alpha, \beta \in \mathcal{L}_{\textbf{D-HT}}$, the proof follows similarly as above, so we leave it to the reader.

Let $\xi = \alpha \to \beta$ where α and β are **D-HT** formulas satisfying induction hypothesis.

$$
\begin{array}{lll}
H \cup T' \in \|\mathsf{tr}(\alpha \to \beta)\|_{\textbf{DL-PA}} & \text{iff} & \text{(see Table 3.4)} \\
H \cup T' \in \|[\mathtt{skip} \cup \mathtt{cpBack}(\mathbb{P}_{\alpha\to\beta})]\Big(\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\Big)\|_{\textbf{DL-PA}} & \text{iff} & \\
H \cup T' \in \|\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} \text{ and} & & \\
H \cup T' \in \|[\mathtt{cpBack}(\mathbb{P}_{\alpha\to\beta})]\Big(\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\Big)\|_{\textbf{DL-PA}} & \text{iff} & \text{(see Lemma 3.1)} \\
H \cup T' \in \|\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} \text{ and} & & \\
\Big(H \cup (T \cap \mathbb{P}_{\alpha\to\beta})\Big) \cup T' \in \|\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} & \text{iff} & \text{(Proposition 3.2)} \\
H \cup T' \in \|\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} \text{ and} & & \\
T \cup T' \in \|\mathsf{tr}(\alpha) \supset \mathsf{tr}(\beta)\|_{\textbf{DL-PA}} & \text{iff} & \text{(by I.H.)} \\
(H,T),(T,T) \in \Big(\mathbb{HT} \setminus \|\alpha\|_{\textbf{D-HT}}\Big) \cup \|\beta\|_{\textbf{D-HT}} & \text{iff} & \text{(see Table 3.1)} \\
(H,T) \in \|\alpha \to \beta\|_{\textbf{D-HT}}. & &
\end{array}
$$

Let $\xi = [\pi]\varphi$ for some $\pi, \varphi \in \mathcal{L}_{\textbf{D-HT}}$. Then we have:

$$
\begin{array}{lll}
(H,T) \in \|[\pi]\varphi\|_{\textbf{D-HT}} & \text{iff} & \text{(see Table 3.1)} \\
\text{for every } (H_1,T_1) \text{ such that } \Big((H,T),(H_1,T_1)\Big) \in \|\pi\|_{\textbf{D-HT}}, & & \\
(H_1,T_1) \in \|\varphi\|_{\textbf{D-HT}} & \text{iff} & \text{(by I.H.)} \\
\text{for every } H_1 \subseteq T_1 \subseteq \mathbb{P} \text{ such that} & & \\
\Big(H \cup T', H_1 \cup T_1'\Big) \in \|\mathsf{tr}(\pi)\|_{\textbf{DL-PA}}, H_1 \cup T_1' \in \|\mathsf{tr}(\varphi)\|_{\textbf{DL-PA}} & \text{iff} & \\
H \cup T' \in \|[\mathsf{tr}(\pi)]\mathsf{tr}(\varphi)\|_{\textbf{DL-PA}} & \text{iff} & \\
H \cup T' \in \|\mathsf{tr}([\pi]\varphi)\|_{\textbf{DL-PA}} & &
\end{array}
$$

The proof of $\xi = \langle\pi\rangle\varphi$ for some $\pi, \varphi \in \mathcal{L}_{\textbf{D-HT}}$ is given through a similar argument, so we leave it to the reader.
Case 2: let ξ be a program of length n.
Let $\xi = \pi_1;\pi_2$ for some $\pi_1, \pi_2 \in \mathcal{L}_{\textbf{D-HT}}$. Then we get the result as follows:

$$
\begin{array}{lll}
\Big((H_1,T_1),(H_2,T_2)\Big) \in \|\pi_1;\pi_2\|_{\textbf{D-HT}} & \text{iff} & \text{(see Table 3.1)} \\
\Big((H_1,T_1),(H_2,T_2)\Big) \in \|\pi_1\|_{\textbf{D-HT}} \circ \|\pi_2\|_{\textbf{D-HT}} & \text{iff} & \\
\text{there is a } \textbf{D-HT} \text{ model } (H_0,T_0) \text{ such that} & & \\
\Big((H_1,T_1),(H_0,T_0)\Big) \in \|\pi_1\|_{\textbf{D-HT}} \text{ and} & & \\
\Big((H_0,T_0),(H_2,T_2)\Big) \in \|\pi_2\|_{\textbf{D-HT}} & \text{iff} & \text{(by I.H.)} \\
\text{there is a valuation } H_0 \cup T_0' \text{ such that} & & \\
\Big(H_1 \cup T_1', H_0 \cup T_0'\Big) \in \|\mathsf{tr}(\pi_1)\|_{\textbf{DL-PA}} \text{ and} & & \\
\Big(H_0 \cup T_0', H_2 \cup T_2'\Big) \in \|\mathsf{tr}(\pi_2)\|_{\textbf{DL-PA}} & \text{iff} & \\
\Big(H_1 \cup T_1', H_2 \cup T_2'\Big) \in \|\mathsf{tr}(\pi_1)\|_{\textbf{DL-PA}} \circ \|\mathsf{tr}(\pi_2)\|_{\textbf{DL-PA}} & \text{iff} & \text{(see Table 3.2)} \\
\Big(H_1 \cup T_1', H_2 \cup T_2'\Big) \in \|\mathsf{tr}(\pi_1);\mathsf{tr}(\pi_2)\|_{\textbf{DL-PA}} & \text{iff} & \\
\Big(H_1 \cup T_1', H_2 \cup T_2'\Big) \cup \in \|\mathsf{tr}(\pi_1;\pi_2)\|_{\textbf{DL-PA}}. & &
\end{array}
$$

Let $\xi = \pi_1 \cup \pi_2$ for some **D-HT** programs π_1 and π_2. Then we have:

$$
\begin{array}{lll}
\Big((H_1, T_1), (H_2, T_2)\Big) \in \|\pi_1 \cup \pi_2\|_{\textbf{D-HT}} & \text{iff} & \text{(Table 3.1)} \\
\Big((H_1, T_1), (H_2, T_2)\Big) \in \|\pi_1\|_{\textbf{D-HT}} \cup \|\pi_2\|_{\textbf{D-HT}} & \text{iff} & \text{(by I.H.)} \\
\Big(H_1 \cup T_1', H_2 \cup T_2'\Big) \|\mathsf{tr}(\pi_1)\|_{\textbf{DL-PA}} \cup \|\mathsf{tr}(\pi_2)\|_{\textbf{DL-PA}} & \text{iff} & \text{(Table 3.2)} \\
\Big(H_1 \cup T_1', H_2 \cup T_2'\Big) \in \|\mathsf{tr}(\pi_1) \cup \mathsf{tr}(\pi_2)\|_{\textbf{DL-PA}} & \text{iff} & \\
\Big(H_1 \cup T_1, H_2 \cup T_2'\Big) \in \|\mathsf{tr}(\pi_1 \cup \pi_2)\|_{\textbf{DL-PA}}. &&
\end{array}
$$

Let $\xi = \pi^*$ where π is a **D-HT** program of length $|\pi| = n - 1$, so π satisfies induction hypothesis. Then, we have the following:

$$
\begin{array}{lll}
\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\pi^*\|_{\textbf{D-HT}} & \text{iff} & \text{(Table 3.1)} \\
\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\pi\|^*_{\textbf{D-HT}} & \text{iff} & \text{(Footnote 1)} \\
\Big((H_i, T_i), (H_j, T_j)\Big) \in \Big(\bigcup_{n>0} \|\pi\|^n_{\textbf{D-HT}}\Big) \cup \|\pi\|^0_{\textbf{D-HT}} & \text{iff} & \text{(Footnote 1)} \\
\text{there is a sequence } (H_i, T_i) = (H_0, T_0), \ldots, (H_n, T_n) = (H_j, T_j) & & \\
\text{of } \textbf{D-HT} \text{ models } (n > 0) \text{ such that for each k, } 0 \leq k < n & & \\
\Big((H_k, T_k), (H_{k+1}, T_{k+1})\Big) \in \|\pi\|_{\textbf{D-HT}} \text{ or } (H_i, T_i) = (H_j, T_j) & \text{iff} & \text{(by I.H.)} \\
\text{there is a sequence } H_i \cup T_i' = H_0 \cup T_0', \ldots, H_n \cup T_n' = H_j \cup T_j' & & \\
\text{of valuations } (n > 0) \text{ such that for each k, } 0 \leq k < n & & \\
\Big(H_k \cup T_k', H_{k+1} \cup T_{k+1}'\Big) \in \|\mathsf{tr}(\pi)\|_{\textbf{DL-PA}} \text{ or } H_i \cup T_i' = H_j \cup T_j' & \text{iff} & \text{(Footnote 1)} \\
\Big(H_i \cup T_i', H_j \cup T_j'\Big) \in \|\mathsf{tr}(\pi)\|^+_{\textbf{DL-PA}} \cup \|\mathsf{tr}(\pi)\|^0_{\textbf{DL-PA}} & \text{iff} & \text{(Footnote 1)} \\
\Big(H_i \cup T_i', H_j \cup T_j'\Big) \in \|\mathsf{tr}(\pi)\|^*_{\textbf{DL-PA}} & \text{iff} & \text{(Table 3.2)} \\
\Big(H_i \cup T_i', H_j \cup T_j'\Big) \in \|\Big(\mathsf{tr}(\pi)\Big)^*\|_{\textbf{DL-PA}} & \text{iff} & \\
\Big(H_i \cup T_i', H_j \cup T_j'\Big) \in \|\mathsf{tr}(\pi^*)\|_{\textbf{DL-PA}}. &&
\end{array}
$$

Let $\xi = \varphi?$ for a **D-HT** formula φ with $|\varphi| = n - 1$. Then, we have:

$$
\begin{array}{lll}
\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\varphi?\|_{\textbf{D-HT}} & \text{iff} & \text{(see Table 3.1)} \\
(H_i, T_i) = (H_j, T_j) \text{ and } (H_i, T_i) \in \|\varphi\|_{\textbf{D-HT}} & \text{iff} & \text{(by I.H.)} \\
H_i \cup T_i' = H_j \cup T_j' \text{ and } H_i \cup T_i' \in \|\mathsf{tr}(\varphi)\|_{\textbf{DL-PA}} & \text{iff} & \text{(see Table 3.2)} \\
\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\mathsf{tr}(\varphi)?\|_{\textbf{DL-PA}} & \text{iff} & \\
\Big((H_i, T_i), (H_j, T_j)\Big) \in \|\mathsf{tr}(\varphi?)\|_{\textbf{DL-PA}}. &&
\end{array}
$$

Hence, $\Pi(\xi)$ holds also for **D-HT** expressions of length n. As a result, $\Pi(\xi)$ is true for all **D-HT** expressions, so we are done. q.e.d.

Theorem A.4 *For every* $\mathcal{L}_{\text{D-HT}}$ *formula* χ, $T \subseteq \mathbb{P}$ *is an equilibrium model of* χ *if and only if* $T \cup T'$ *is a* **DL-PA** *model of* $\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$.

Proof. $T \cup T'$ is a **DL-PA** model of $\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$ if and only if

(A.1) $\qquad T \cup T'$ is a **DL-PA** model of $\mathsf{tr}(\chi)$

and

(A.2) $\qquad T \cup T'$ is a **DL-PA** model of ${\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$

By the Main Lemma, (A.1) is the case if and only if (T, T) is a HT model of χ in **D-HT**. It remains to prove that (A.2) is the case if and only if (H, T) is not a HT model of χ, for any set $H \subset T$. We establish this by proving that the following statements are equivalent.

1. $T \cup T'$ is a **DL-PA** model of ${\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$
2. $(T \cap \mathbb{P}_\chi) \cup T'$ is not a **DL-PA** model of $\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$ (Proposition 3.2)
3. $H \cup T'$ is not a **DL-PA** model of $\mathsf{tr}\mathbb{P}_\chi(\chi)$, for any $H \subset T \cap \mathbb{P}_\chi$ (Program Lemma 3.1)
4. H, T is not a HT model of χ, for any set $H \subset T \cap \mathbb{P}_\chi$ (Main Lemma 3.2)
5. H, T is not a HT model of χ, for any set $H \subset T$ (Proposition 3.1).

q.e.d.

Theorem A.5 *Let* χ *and* φ *be* $\mathcal{L}_{\text{D-HT}}$ *formulas. Then* $\chi \mid\!\approx \varphi$ *if and only if*

$$\langle \mathsf{cp}(\mathbb{P}_\chi \cup \mathbb{P}_\varphi)\rangle \Big(\big(\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}(\chi) \big) \supset \mathsf{tr}(\varphi) \Big)$$

is **DL-PA** *valid.*

Proof. As the program $\mathsf{cp}(\mathbb{P}_\chi)$ is both deterministic and always executable, T is a **DL-PA** model of $\langle \mathsf{cp}(\mathbb{P}_\chi)\rangle \, (\mathsf{tr}(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi))$ if and only if xxx

(A.3) $\qquad T$ is a **DL-PA** model of $\langle \mathsf{cp}(\mathbb{P}_\chi)\rangle {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$

(Remember that we consider that $T = T \cup \emptyset'$, i.e., the set of copied variables is empty.)

By the Program Lemma 3.1, T is a **DL-PA** model of $\langle \mathtt{cp}(\mathbb{P}_\chi)\rangle \mathsf{tr}(\chi)$ if and only if $T \cup (T \cap \mathbb{P}_\chi)'$ is a **DL-PA** model of $\mathsf{tr}(\chi)$, and by Proposition 3.1, the latter is the case if and only if $T \cup T'$ is a **DL-PA** model of $\mathsf{tr}(\chi)$; finally, by the Main Lemma 3.2, the latter is the case if and only if (T, T) is a HT model of χ in **D-HT**.

It remains to prove that (A.3) is the case if and only if (H, T) is not a HT model of χ, for any set $H \subset T$. We establish this by proving that the following statements are equivalent.

1. T is a **DL-PA** model of $\langle \mathtt{cp}(\mathbb{P}_\chi)\rangle \sim \langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$
2. $T \cup (T \cap \mathbb{P}_\chi)'$ is a **DL-PA** model of $\sim \langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$ (Program Lemma 3.1)
3. $(T \cap \mathbb{P}_\chi) \cup T'$ is not a **DL-PA** model of $\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle \mathsf{tr}\mathbb{P}_\chi(\chi)$ (Proposition 3.2, twice)
4. $H \cup T'$ is not a **DL-PA** model of $\mathsf{tr}\mathbb{P}_\chi(\chi)$, for any $H \subset T \cap \mathbb{P}_\chi$ (Program Lemma 3.1)
5. H, T is not a HT model of χ, for any set $H \subset T \cap \mathbb{P}_\chi$ (Main Lemma 3.2)
6. H, T is not a HT model of χ, for any set $H \subset T$ (Proposition 3.1).

q.e.d.

A.4 Proofs of Chapter 4

Proposition A.10 *Let* $\big((\mathcal{T}, \hbar), T\big)$ *be an EHT model. Let* φ *be an* $\mathcal{L}_{\text{E-HT}}$ *formula. Then:*

1. $(\mathcal{T}, \hbar), T \models_{\text{E-HT}} \neg\varphi$ *if* $(\mathcal{T}, id), T \not\models_{\text{E-HT}} \varphi$;
2. $(\mathcal{T}, \hbar), T \models_{\text{E-HT}} \neg\neg\varphi$ *ff* $(\mathcal{T}, id), T \models_{\text{E-HT}} \varphi$;
3. $(\mathcal{T}, \hbar), T \models_{\text{E-HT}} \neg \mathsf{K}\,\varphi$ *if* $(\mathcal{T}, id), T' \not\models_{\text{E-HT}} \varphi$, *for some* $T' \in \mathcal{T}$;
4. $(\mathcal{T}, \hbar), T \models_{\text{E-HT}} \neg \hat{\mathsf{K}}\,\varphi$ *if* $(\mathcal{T}, id), T' \not\models_{\text{E-HT}} \varphi$, *for any* $T' \in \mathcal{T}$.

Proof.

- The proof of item 1 is given by using Lemma 4.2. One side of the proof is trivial, so we leave it to the reader. The other side immediately follows from the heredity property of **E-HT** logic (see Lemma 4.2) as in:

$$\begin{array}{rcl}(\mathcal{T},id),T_0 \not\models_{\textbf{E-HT}} \varphi & \Longrightarrow & (\mathcal{T},id),T_0 \not\models_{\textbf{E-HT}} \varphi \text{ and } (\mathcal{T},\hbar),T_0 \not\models_{\textbf{E-HT}} \varphi \\ & \Longrightarrow & (\mathcal{T},\hbar),T_0 \models_{\textbf{E-HT}} \neg\varphi.\end{array}$$

- Item 2 is an easy consequence of Lemma 1.2.1 (item 1). So, we have:

$$\begin{array}{lll}(\mathcal{T},\hbar),T_0 \models_{\textbf{E-HT}} \neg\neg\varphi & \text{iff} & (\mathcal{T},id),T_0 \not\models_{\textbf{E-HT}} \neg\varphi \text{ (replace } \varphi \text{ by } \neg\varphi \text{ in item 1)} \\ & \text{iff} & (\mathcal{T},id),T_0 \models_{\textbf{E-HT}} \varphi.\end{array}$$

- The proof of item 3 also follows from Lemma 1.2.1 (item 1) through the interpretation of K operator (see "Truth conditions" in Subsection 4.1.2) described below:

$$\begin{array}{lll}(\mathcal{T},\hbar),T_0 \models_{\textbf{E-HT}} \neg\mathsf{K}\varphi & \text{iff} & (\mathcal{T},id),T_0 \not\models_{\textbf{E-HT}} \mathsf{K}\varphi \text{ (update } \varphi \text{ by } \mathsf{K}\varphi \text{ in item 1)} \\ & \text{iff} & (\mathcal{T},id),T' \not\models_{\textbf{E-HT}} \varphi \text{ for some } T' \in \mathcal{T}.\end{array}$$

- We prove item 4 again by means of Lemma 1.2.1 (item 1), using the interpretation of $\hat{\mathsf{K}}$ operator (see "Truth conditions" in Subsection 4.1.2) as follows:

$$\begin{array}{lll}(\mathcal{T},\hbar),T_0 \models_{\textbf{E-HT}} \neg\hat{\mathsf{K}}\varphi & \text{iff} & (\mathcal{T},id),T_0 \not\models_{\textbf{E-HT}} \hat{\mathsf{K}}\varphi \text{ (use } \varphi{:=}\hat{\mathsf{K}}\varphi \text{ in item 1)} \\ & \text{iff} & (\mathcal{T},id),T' \not\models_{\textbf{E-HT}} \varphi \text{ for any } T' \in \mathcal{T}.\end{array}$$

q.e.d.

Proposition A.11 *The following properties hold for* $\varphi \in \mathcal{L}_{\textbf{E-HT}}$.

1. *All EHT valid formulas have exactly one EEM, namely* $\{\emptyset\}$.
2. $\neg\varphi$ *has either a unique EEM* $\{\emptyset\}$ *or none: if* $\{\emptyset\} \models_{\textbf{S5}} \varphi$ *then* $\{\emptyset\}$ *is the unique EEM of* $\neg\neg\varphi$ *and* $\neg\varphi$ *has none, otherwise vice versa.*
3. *If* φ *has no EEM then neither do* $\hat{\mathsf{K}}\varphi$ *and* $\mathsf{K}\varphi$.
4. *Let* $T \subseteq \mathbb{P}$ *be a valuation. For an objective (nonmodal)* φ *(i.e.,* $\varphi \in \mathcal{L}_{\textbf{HT}}$*),* $\{T\}$ *and* $\{\emptyset, \underline{T}\}$ *are EEMs of* φ *if and only if* T *is an EM of* φ.
5. *For an objective* φ*, any arbitrary collection of EMs of* φ *is an EEM of* $\mathsf{K}\varphi$.

6. *For a nonmodal $\mathcal{L}_{\text{E-HT}}$ formula φ, T is an EM of φ iff:*
   (a) *$\{T\}$ is an EEM of $\hat{\mathsf{K}}\varphi$ when $\emptyset \models \varphi$.*
   (b) *$\{T\}$ and $\{\emptyset, T\}$ are the EEMs of $\hat{\mathsf{K}}\varphi$ when $\emptyset \not\models \varphi$.*

Proof.

- the proof of item 2: remember that $(\mathcal{T}, \hbar), T \models_{\text{E-HT}} \neg\varphi$ if and only if $(\mathcal{T}, id), T \not\models_{\text{E-HT}} \varphi$ (see Proposition A.10). This means that the two conditions of epistemic equilibrium model of $\neg\varphi$ in fact amount to the same condition. Then, it follows immediately that the only epistemic equilibrium model candidate for $\neg\varphi$ is $\{\emptyset\}$ where the second minimality condition is trivially satisfied. However, when $\{\emptyset\} \models_{\text{S5}} \varphi$ then $\{\emptyset\} \not\models_{\text{S5}} \neg\varphi$, so $\{\emptyset\}$ is not an epistemic equilibrium model of $\neg\varphi$. On the other hand, when $\{\emptyset\} \not\models_{\text{S5}} \varphi$ then it follows that $\{\emptyset\} \models_{\text{S5}} \varphi$, so $\{\emptyset\}$ happens to be an epistemic equilibrium model of $\neg\varphi$.

- the proof of item 3: assume that φ doesn't have an epistemic equilibrium model $(*)$.

  – Assume for a contradiction that the $\mathcal{L}_{\text{E-HT}}$ formula $\hat{\mathsf{K}}\varphi$ has an epistemic equilibrium model, say $\big(\mathcal{T}, T_0\big)$ $(**)$. Hence, by definition of epistemic equilibrium model (item 1), we have $(\mathcal{T}, id), T_0 \models_{\text{E-HT}} \hat{\mathsf{K}}\varphi$. Then, by "Truth conditions" in Subsection 4.1.2, we get $(\mathcal{T}, id), T' \models_{\text{E-HT}} \varphi$ for some $T' \in \mathcal{T}$, but then using the assumption $(*)$, we conclude that there is a strictly weaker model $\big((\mathcal{T}, \hbar), T'\big)$ of $\big((\mathcal{T}, id), T'\big)$ such that $(\mathcal{T}, \hbar), T' \models_{\text{E-HT}} \varphi$ $(\bullet)$; otherwise that would be an epistemic equilibrium model of φ. However, from $(**)$ using the minimality condition, we obtain that $(\mathcal{T}, \hbar), T_0 \not\models_{\text{E-HT}} \hat{\mathsf{K}}\varphi$ for any strictly weaker model $\big((\mathcal{T}, \hbar), T_0\big)$ of $\big((\mathcal{T}, id), T_0\big)$ which further gives us $(\mathcal{T}, \hbar), T \not\models_{\text{E-HT}} \varphi$ for any $T \in \mathcal{T}$. Since the latter result contradicts $(\bullet)$, we conclude that $\hat{\mathsf{K}}\varphi$ has no epistemic equilibrium model either.

  – The proof for $\mathsf{K}\varphi$ follows more or less the same, so we leave it to the reader.

q.e.d.

Proposition A.12 *The following properties hold: for $\varphi, \psi \in \mathcal{L}_{\text{HT}}$,*

1. *if $\varphi \mid\!\approx \psi$ then $\mathsf{K}\varphi \mid\!\approx \mathsf{K}\psi$ and $\mathsf{K}\varphi \mid\!\approx \psi$.*

2. *if* $\varphi \mid\!\approx \psi$ *then* $\hat{\mathsf{K}}\varphi \mid\!\approx \hat{\mathsf{K}}\psi$ *and* $\varphi \mid\!\approx \hat{\mathsf{K}}\psi$.

PROOF.

1. Assume that $\varphi \mid\!\approx \psi$ holds in equilibrium logic $(*)$. Let $\mathcal{T}$ be an epistemic equilibrium model of $\mathsf{K}\varphi$. Then, by Lemma **??.??**, T is an equilibrium model of φ for every $T \in \mathcal{T}$. Hence, from $(*)$ we obtain that $(T,T) \models_{\mathbf{HT}} \psi$ for every $T \in \mathcal{T}$. As a result, $(\mathcal{T}, id) \models_{\mathbf{E\text{-}HT}} \mathsf{K}\psi$. As to the second part the proof, we suppose $\left(\mathcal{T}, T_0\right)$ is an epistemic equilibrium model of $\mathsf{K}\varphi$. Then, by the same reasoning, we get $(T,T) \models_{\mathbf{HT}} \psi$ for every $T \in \mathcal{T}$. Hence, so does (T_0,T_0). Therefore, $(\mathcal{T}, id), T_0 \models_{\mathbf{E\text{-}HT}} \psi$.

2. Assume $\varphi \mid\!\approx \psi$ in equilibrium logic $(*)$. Let $\mathcal{T}$ be an epistemic equilibrium model of $\hat{\mathsf{K}}\varphi$. Then, by Lemma **??.??**, $\mathcal{T}$ is either in the form of $\{T\}$ or in the form of $\left\{\emptyset, T\right\}$ where T is an equilibrium model of φ. Hence, from $(*)$ we obtain that $(T,T) \models_{\mathbf{HT}} \psi$, but then $(\mathcal{T}, id) \models_{\mathbf{E\text{-}HT}} \hat{\mathsf{K}}\psi$ immediately follows. As for the verification of $\varphi \mid\!\approx \hat{\mathsf{K}}\psi$, we first take an arbitrary epistemic equilibrium model of φ, say $\left(\mathcal{T}, T_0\right)$. Then, by Lemma **??**, $\left(\mathcal{T}, T_0\right)$ is either in the form T or in the form $\left\{\emptyset, \underline{T}\right\}$ where T is an equilibrium model of φ. The rest follows by the same reasoning as the first part.

q.e.d.

Theorem A.6 *Let* Φ *and* Ψ *be two* **E-HT** *theories. The following are equivalent:*

1. $\|\Phi\|_{\mathbf{E\text{-}HT}} = \|\Psi\|_{\mathbf{E\text{-}HT}}$*;*

2. $\Phi \cup \Theta$ *and* $\Psi \cup \Theta$ *have the same epistemic equilibrium models, for every* Θ.

PROOF. The left-to-right (1⇒2) direction is straightforward: if no EHT model allows to distinguish Φ and Ψ then there is no EHT model distinguishing $\Phi \cup \Theta$ and $\Psi \cup \Theta$, whatever Θ is, and *a fortiori* there is no such **S5** model.

The rest of the proof is devoted to the right-to-left (2⇒1) direction. Assume that $\|\Phi\|_{\mathbf{E\text{-}HT}} \neq \|\Psi\|_{\mathbf{E\text{-}HT}}$. Then there is an (pointed) EHT model, say $\left((\mathcal{T}, \hbar), T_0\right)$, which is a model of Φ but not of Ψ or vice versa. Without loss of generality, let $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Phi$ and $(\mathcal{T}, \hbar), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi$. We now consider two cases in which we construct a set of formulas Θ such that $(\mathcal{T}, T_0)$ is an EEM of one of $\Phi \cup \Theta$ and $\Psi \cup \Theta$, but not the other. This establishes that the EEMs of $\Phi \cup \Theta$ and $\Psi \cup \Theta$ differ for some Θ.

1. Let $\mathcal{T}, T_0 \not\models_{\mathbf{S5}} \Psi$. We here construct an EHT theory

$$\Theta = \left\{ \hat{\mathsf{K}}\, \chi_T \ : \ T \in \mathcal{T} \right\}.$$

Clearly, $\mathcal{T}, T_0 \not\models_{\mathbf{S5}} \Psi \cup \Theta$, which further means that $(\mathcal{T}, T_0)$ is not an EEM of $\Psi \cup \Theta$. The rest is to prove that $(\mathcal{T}, T_0)$ is an EEM of $\Phi \cup \Theta$.
The first step is easy: since $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Phi$ by hypothesis, $(\mathcal{T}, id), T_0 \models_{\mathbf{E\text{-}HT}} \Phi$ by heredity (see Proposition 4.2). This further means that $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Phi$. Moreover, thanks to the construction of Θ, we also have $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Phi \cup \Theta$.
It remains to examine the minimality condition: let $\hbar' : \mathcal{T} \to 2^{\mathbb{P}}$ be such that $(\mathcal{T}, \hbar') \lhd (\mathcal{T}, id)$. Then $\hbar' \neq id$, and so Lemma 4.1 implies $(\mathcal{T}, \hbar'), T' \not\models_{\mathbf{E\text{-}HT}} \Theta$ for any $T' \in \mathcal{T}$. In particular, $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Theta$ Hence, we conclude that $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Phi \cup \Theta$. Therefore, $(\mathcal{T}, T_0)$ is an EEM of $\Phi \cup \Theta$.

2. Let $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Psi$. We then construct an EHT theory

$$\Theta = \left\{ \hat{\mathsf{K}}\, \eta_{T,\hbar} \ : \ T \in \mathcal{T} \right\}.$$

To begin with, by Lemma 4.1 we have: $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Theta$ if and only if $\hbar = \hbar$ or $\hbar = id$. Therefore, it immediately follows that $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Theta$. However, $\hbar$ cannot be id because we have supposed in this case that $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Psi$ which means that $(\mathcal{T}, id), T_0 \models_{\mathbf{E\text{-}HT}} \Psi$, and by hypothesis we also have $(\mathcal{T}, \hbar), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi$. Again by hypothesis we also have $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Phi$ which further gives us $(\mathcal{T}, \hbar), T_0 \models_{\mathbf{E\text{-}HT}} \Phi \cup \Theta$ for $\hbar \neq id$. As a result, $(\mathcal{T}, T_0)$ cannot be an EEM of $\Phi \cup \Theta$.
The rest is to show that $(\mathcal{T}, T_0)$ is an EEM of $\Psi \cup \Theta$. Using Lemma 4.1, we also have: $(\mathcal{T}, id), T_0 \models_{\mathbf{E\text{-}HT}} \Theta$ if and only if $id = \hbar$ or $id = id$. Hence, it is clear that $(\mathcal{T}, id), T_0 \models_{\mathbf{E\text{-}HT}} \Theta$, which is the same as $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Theta$. Then, we have by our initial assumption that $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Psi \cup \Theta$. We now need to prove the minimality condition: $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi \cup \Theta$ for any $\hbar'$ such that $(\mathcal{T}, \hbar') \trianglelefteq (\mathcal{T}, id)$. It is enough to show that $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi$ or $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Theta$ for any such $\hbar'$. When $(\mathcal{T}, \hbar'), T_0 \not\models_{\mathbf{E\text{-}HT}} \Theta$ the result trivially follows, so assume that $(\mathcal{T}, \hbar'), T_0 \models_{\mathbf{E\text{-}HT}} \Theta$. Since $(\mathcal{T}, \hbar') \trianglelefteq (\mathcal{T}, id)$ we know that $\hbar' \neq id$. Then by Lemma 4.1, $\hbar' = \hbar$. However, by hypothesis, $(\mathcal{T}, \hbar), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi$ which further implies that $(\mathcal{T}, \hbar), T_0 \not\models_{\mathbf{E\text{-}HT}} \Psi \cup \Theta$. As a result, $(\mathcal{T}, T_0)$ is an EEM of $\Psi \cup \Theta$.

This ends the proof of right-to-left direction. q.e.d.

Theorem A.7 *Let Φ and Ψ be two* **E-HT** *theories. The following are equivalent:*

1. $\|\Phi\|_{\textbf{E-HT}} = \|\Psi\|_{\textbf{E-HT}}$;

2. $\Phi \cup \Theta$ *and* $\Psi \cup \Theta$ *have the same autoepistemic equilibrium models, for all* Θ.

PROOF. The proof follows the lines of that of Theorem 4.2 and only the construction of the set Θ has to be adapted in both cases.

For the case where $\mathcal{T}, T_0 \models_{\mathbf{S5}} \Psi$ we define

$$\Theta = \left\{ \hat{\mathsf{K}}\, \chi_T \ : \ T \in \mathcal{T} \right\} \cup \left\{ \mathsf{K} \bigvee_{T \in \mathcal{T}} \chi_T \right\}.$$

This guarantees that no model strictly including $\mathcal{T}$ is an epistemic equilibrium model of $\Phi \cup \Theta$.

Similarly, for the case where $\mathcal{T}, T_0 \not\models_{\mathbf{S5}} \Psi$ we define

$$\Theta = \left\{ \hat{\mathsf{K}}\, \eta_{T,\hbar} \ : \ T \in \mathcal{T} \right\} \cup \left\{ \mathsf{K} \bigvee_{T \in \mathcal{T}} \chi_T \right\}.$$

This guarantees that no model strictly including $\mathcal{T}$ is an epistemic equilibrium model of $\Psi \cup \Theta$. q.e.d.

Appendix B

Preliminary Instructions

B.1 Strong negation

Strong negation was first introduced by Nelson (Nelson [1949]) in order to model a logical concept of constructible falsity, so it is also known as *constructive negation*. He originally presented it as an alternative to intuitionistic negation ($\neg$) which is also in a sense "constructive" since (only) entailed formulas can be constructively proved. Strong negation (due to Nelson in Nelson [1949]) adds to intuitionistic logic the insight that primitive propositions may be not only constructively verified but also constructively falsified.

Strong negation is sometimes called 'explicit' negation or even 'classical' negation in the **ASP** literature. We denote it by $\sim$ in this paper, and name this concept only as strong negation. On the other hand, weak negation corresponds to NAF in the **ASP** literature, and is often denoted by the symbol *not*.

Two essential features characterise the concept of strong negation:

- one can characterise atomic negative information by strong negation.
- there exists an effective procedure to reduce falsification of complex statements to verification or falsification of its parts by a normal form transformation.

The concept of strong negation was first axiomatised by Vorob'ev (Vorob'ev [1952]). Nelson's constructive logic **N** with its additional axioms is a conservative extension of Heyting's intuitionistic logic in the sense that any formula of constructive logic without strong negation in constructive logic is a theorem if and only if it is a theorem in intuitionistic logic. In constructive logic, intuitionistic negation is definable in terms of strong negation by (due to Vorob'ev):

$$\neg\varphi \stackrel{\mathsf{def}}{=} \varphi \rightarrow \sim\varphi.$$

Vorob'ev also showed an interesting property of strong negation (Vorob'ev [1952]): any arbitrary formula including strong negation is equivalent to a formula in *reduced* form, that is, the strong negation is just allowed to appear in front of propositional variables in the form of a negative literal.

Strong negation under Vorob'ev axioms (N1-N6) (see Subsection 1.2.4) has the following interesting property: when (only) these axioms are added to a superintuitionistic logic, one obtains a conservative extension of the system axioms. To express it more clearly, this least extension by strong negation does not change the theorems of the original system, and instead it only adds new theorems involving strong negation. So, strong negation is highly strong.

B.1.1 Representing the negative information using strong negation

Strong negation helps us represent the negative information explicitly in a logic program. In general **LP** negative information is represented implicitly using CWA, but there are cases this implicit representation does not work and the difference between *not* p and $\sim p$ should essentially be specified. A well-known example is given by John McCarthy: a school bus can cross railway tracks unless there is an approaching train which can be expressed by the following **ASP** rule:

$$\text{Cross} \leftarrow \mathit{not}\ \ \text{Train}\ .$$

In general **LP**, the unique stable model of the program containing only this rule is $\{Cross\}$. So, the program is supposed to return an answer *yes* for a query '$Cross$?' and an answer *no* for a query '$Train$?'. However, this knowledge representation is not acceptable due to the fact that the information about the presence or the absence of an approaching train is not available, and hence CWA is actually is not applicable to $Train$. On the other hand, in extended **LP** the same program answers both questions respectively *yes* and *unknown* which reveals the failure more apparently. This failure is just eliminated having *not* being replaced by strong negation in this rule:

$$\text{Cross} \leftarrow \sim \text{Train}\ .$$

In extended **LP**, the only answer set of the program uniquely containing this rule is $\emptyset$. So, the program returns both queries above *unknown*, and the answer set will never include $Cross$ unless $\sim Train$ is included in the database.

Appendix C

Some Forms of Nonmonotonic Reasoning

C.1 Default logic

Default logic is a non-monotonic logic proposed by Raymond Reiter to formalise reasoning with default assumptions. It can express facts like "by default, p is true", in contrast to classical logic, which can only express facts like "p is true" or "p is false". However reasoning often involves facts that are true in the majority of cases but not always. The review of default logic below is restricted to the case of ground defaults which is sufficient for our purposes.

Following Gelfond and Lifschitz's notation, we identify a *default* d an expression of the form

$$F \leftarrow G \ : \ \mathrm{M}H_1, \ldots, \mathrm{M}H_k \tag{C.1}$$

where F, G, H_i are propositional formulas for $1 \leq i \leq k$ such that $k \geq 0$. We call F and G respectively the *consequent* and the *prerequisite* (or *precondition*) of the default, and the H_i's are its *justifications*. However, according to Reiter's notation (Reiter [1980]), Definition C.1 can be equivalently represented by

$$\frac{G \ : \ \mathrm{M}H_1, \ldots, \mathrm{M}H_k}{F} \quad \text{or} \quad G \ : \ \mathrm{M}H_1, \ldots, \mathrm{M}H_k \ / \ F.$$

So, an axiom F can be identified with the default '$F \leftarrow \mathit{true} \ :$'. Defaults are generalisations of inference rules. A default theory D is a finite set of defaults.

C.2 Minimal belief and negation as failure

The logic of *minimal belief and negation as failure* (**MB-NF**) was proposed by Lifschitz as a general nonmonotonic logic (Lifschitz [1994]). It has two modal

operators: the NAF operator *not* and the B operator characterising minimal belief. It comprises the class of logic programs that allows for both the NAF operator and strong negation. It is one of the most expressive logics and can serve as a common framework unifying several nonmonotonic formalisms. However, **MB-NF** is purely semantical and too intractable to be used directly for representing knowledge.

Appendix D

Propositional Dynamic Logic

D.1 Syntax

The language of **PDL** has expressions of two sorts: formulas $\varphi, \psi \ldots$ and programs $\pi_1, \pi_2, \ldots$. They are built inductively from the atomic ones using the propositional operators $\supset$ and $\bot$, the program operators $\cup$, $(\ )^*$, ; and the mixed operators $(\ )?$ and $[\ ]$. Compound programs and propositions of **PDL** have the following intuitive meanings:

- *Atomic* programs are basic, and they execute in a single step. They are called atomic because they are indivisible, i.e., they cannot be decomposed any further.
- The operator '(.)?' is the *test* operator, so the program '$\varphi?$' checks whether the property φ holds in the current state. If it holds, it proceeds without changing the current state. If not, it blocks without halting (fails).
- The operator ';' is the *sequential composition* operator. The program '$\pi_1; \pi_2$' means: "first execute π_1, and then execute π_2".
- The operator '$\cup$' is the *nondeterministic composition* (*choice*) operator. The program '$\pi_1 \cup \pi_2$' means: "nondeterministically choose one of π_1 or π_2 and execute it".
- The operator '$(.)^*$' is the (finite) *iteration* operator. It may also be called *Kleene star*. The program '$(\pi)^*$' means: "execute π some nondeterministically chosen finite number of times (zero or more)".
- '$[\pi]\varphi$' means: "it is necessary that after executing π, φ is true".

The possibility operator $\langle\ \rangle$ is the modal dual of the necessity operator $[\ ]$, so it is defined by

$$\langle\pi\rangle\varphi \stackrel{\text{def}}{=} \sim[\pi]\sim\varphi.$$

$\langle\pi\rangle\varphi$ has the following intuitive meaning: "there is a computation of π that terminates in a state satisfying φ". One important difference between $\langle\ \rangle$ and $[\ ]$ is that while the former terminates, the latter does not. Indeed, the formula $[\pi]\bot$ asserts that no computation of π terminates, and the formula $[\pi]\top$ is always true, regardless of π.

The operators of deterministic *while* programs can be defined in terms of the atomic operators:

$$\text{if } \varphi \text{ then } \pi_1 \text{ else } \pi_2 \stackrel{\text{def}}{=} (\varphi?;\pi_1) \cup (\sim\varphi?;\pi_2)$$
$$\text{while } \varphi \text{ do } \psi \stackrel{\text{def}}{=} (\varphi?;\pi)^*;\sim\varphi?$$

The latter has the following explicit form

$$\underbrace{(\varphi?;\pi)\,;\,\ldots\,;\,(\varphi?;\pi)}_{n};\sim\varphi$$

where $\pi^0 \stackrel{\text{def}}{=} \texttt{skip}$ and $\pi^{n+1} = \pi^n;\pi$ for $0 \leq n < \infty$. The programs `skip` and `fail` do nothing. They are respectively the *no-operation* and the *failing* programs. In addition to the previous definition of `skip` as $\pi^0 \stackrel{\text{def}}{=} \texttt{skip}$, these trivial programs can be defined from the atomic programs as follows: $\texttt{skip} \stackrel{\text{def}}{=} \top?$ and $\texttt{fail} \stackrel{\text{def}}{=} \bot?$

The semantics of **PDL** is given in a similar manner as with **DL-PA**, so we do not include it here. However, for a detailed overview, one can refer Harel et al. [2000]. In the following section, we give the axiomatisation of **PDL**.

D.2 A deductive system

The axiom schemas and the inference rules are listed in Table D.1. They constitutes a sound and complete Hilbert-style deductive system for **PDL**.

The second and third axiom schemas as well as the inference rules in Table D.1 are not particular to **PDL**. They come from modal logic (see Blackburn et al. [2001b]; Chellas [1980]; Hughes and Cresswell [2012]). The latter inference rule is also known as *necessitation*. The axiom schema $\varphi \wedge [\pi^*](\varphi \supset [\pi]\varphi) \supset [\pi^*]\varphi$ is called the PDL *induction* axiom . Intuitively, it says: "suppose φ is true in the current state, and also suppose that after any number of iterations of π if φ is still true, then it will be true after one more iteration of π. Then φ will be true after any number of iterations of π". In other words, "if φ is true initially, and if the truth of φ is preserved by the program π, then φ will be true after any number of iterations of π".

Axioms for propositional logic

$[\pi](\varphi \supset \psi) \supset ([\pi]\varphi \supset [\pi]\psi)$

$[\pi](\varphi \wedge \psi) \equiv ([\pi]\varphi \wedge [\pi]\psi)$

$[\pi_1 \cup \pi_2]\varphi \equiv [\pi_1]\varphi \wedge [\pi_2]\varphi$

$[\pi_1; \pi_2]\varphi \equiv [\pi_1][\pi_2]\varphi$

$[\varphi?]\psi \equiv \varphi \supset \psi_1$

$\varphi \wedge [\pi][\pi^*]\varphi \equiv [\pi^*]\varphi$

$\varphi \wedge [\pi^*](\varphi \supset [\pi]\varphi) \supset [\pi^*]\varphi$ (induction axiom)

(Modus ponens) $\dfrac{\varphi, \quad \varphi \supset \psi}{\psi}$

(Generalisation) $\dfrac{\varphi}{[\pi]\varphi}$

Table D.1: Axiomatisation of **PDL**

References

Felicidad Aguado, Pedro Cabalar, Gilberto Pérez, and Concepción Vidal. Strongly equivalent temporal logic programs. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *JELIA*, volume 5293 of *Lecture Notes in Computer Science*, pages 8–20. Springer, 2008. ISBN 978-3-540-87802-5.

Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

Krzysztof R. Apt and Marc Bezem. Acyclic programs. *New generation computing*, 9(3-4):335–363, 1991.

Leo Bachmair and Harald Ganzinger. Perfect model semantics for logic programs with equality. In *ICLP*, pages 645–659, 1991.

Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In Orna Kupferman, editor, *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 143–152, http://www.ieee.org/, Juin 2013. IEEE Computer Society, IEEE.

Chitta Baral. *Knowledge representation, reasoning and declarative problem solving.* Cambridge University Press, 2003.

Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *The Journal of Logic Programming*, 19:73–148, 1994.

Gavin M. Bierman and Valeria de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65(3):383–416, 2000.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001a.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic.* Cambridge Tracts in Theoretical Computer Science. University Press, 2001b.

Gerhard Brewka, Salem Benferhat, and Daniel Le Berre. Qualitative choice logic. *Artificial Intelligence*, 157(1):203–237, 2004a.

Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Logic programs with ordered disjunction. *Computational Intelligence*, 20(2):335–357, 2004b.

Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors. *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, 2012. AAAI Press. ISBN 978-1-57735-560-1.

Pedro Cabalar. A logical charaterisation of ordered disjunction. *CoRR*, abs/1011.4833, 2010.

Pedro Cabalar and Stéphane Demri. Automata-based computation of temporal equilibrium models. In Germán Vidal, editor, *LOPSTR*, volume 7225 of *Lecture Notes in Computer Science*, pages 57–72. Springer, 2011. ISBN 978-3-642-32210-5.

Pedro Cabalar and Paolo Ferraris. Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming (TPLP)*, 7(6): 745–759, 2007.

Pedro Cabalar, David Pearce, and Agustín Valverde. Minimal logic programs. In Verónica Dahl and Ilkka Niemelä, editors, *Proc. ICLP*, volume 4670 of *LNCS*, pages 104–118. Springer Verlag, 2007. ISBN 978-3-540-74608-9.

Walter A. Carnielli, Claudio Pizzi, and Juliana Bueno-Soler. *Modalities and multimodalities*. Logic, Epistemology, and the Unity of Science. Springer Verlag, 2009.

Brian F. Chellas. *Modal logic: an introduction*. Cambridge University Press, 1980.

Jianhua Chen. Minimal knowledge + negation as failure = only knowing (sometimes). In *LPNMR*, pages 132–150, 1993.

Jianhua Chen. The logic of only knowing as a unified framework for non-monotonic reasoning. *Fundamenta Informaticae*, 21(3):205–220, 1994.

Jianhua Chen. The generalized logic of only knowing (gol) that covers the notion of epistemic specifications. *Journal of Logic and Computation*, 7(2):159–174, 1997.

Keith L. Clark. Negation as failure. In *Logic and Databases*, pages 293–322. Springer, 1978.

Martin Diéguez Lodeiro. *Temporal answer set programming*. Phd thesis, University of Corunna, Computer Science Department, Facultade de Informática, February 2015.

Thomas Eiter. Answer set programming in a nutshell. In *Workshop Freiburg*, volume 2, page 50. Citeseer, 2008.

Thomas Eiter and Georg Gottlob. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In *ILPS*, pages 266–278. Citeseer, 1993.

Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits. Using methods of declarative logic programming for intelligent information agents. *TPLP*, 2(6):645–709, 2002.

Patrice Enjalbert and Luis Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

Esra Erdem and Vladimir Lifschitz. Transformations of logic programs related to causality and planning. In *Logic Programming and Nonmonotonic Reasoning*, pages 107–116. Springer, 1999.

Luis Fariñas del Cerro and Andreas. Herzig. Modal deduction with applications in epistemic and temporal logic. In Dov Gabbay, J. Chris, and J. A. Robinson, editors, *Handbook of Logic and Artificial Intelligence*, volume 4, pages 499–594. Oxford, 1995. (Epistemic and Temporal Reasoning).

Luis Fariñas del Cerro and Andreas Herzig. The modal logic of equilibrium models. In *Frontiers of Combining Systems (FroCoS)*, pages 135–146, http://www.springerlink.com, 2011a. Springer Verlag. URL http://www.irit.fr/~Andreas.Herzig/P/Frocos11.html.

Luis Fariñas del Cerro and Andreas Herzig. Contingency-based equilibrium logic. In *Logic Programming and Nonmonotonic Reasoning*, pages 223–228, http://www.springerlink.com, 2011b. Springer Verlag. URL http://www.irit.fr/~Andreas.Herzig/P/lpnmr2011.html.

Luis Fariñas del Cerro and Andrés R. Raggio. Some results in intuitionistic modal logic. *Logique et Analyse Louvain*, 26(102):219–224, 1983.

Luis Fariñas del Cerro, David Fauthoux, Olivier Gasquet, Andreas Herzig, Dominique Longin, and Fabio. Massacci. Lotrec : The generic tableau prover for modal and description logics. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *First International Joint Conference on Automated Reasoning*

(IJCAR 2001), Siena, Italy, June 18-23, 2001, Proceedings, volume 2083 of *Lecture Notes in Computer Science (LNCS)*, pages 453–458. Springer-Verlag, 2001. ISBN 3-540-42254-4.

Luis Fariñas del Cerro, Andreas Herzig, and Ezgi Iraz Su. Combining equilibrium logic and dynamic logic. In Pedro Cabalar and Tran Cao Son, editors, *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *Lecture Notes in Computer Science*, pages 304–316. Springer, 2013. ISBN 978-3-642-40563-1, 978-3-642-40564-8.

Paolo Ferraris. Answer sets for propositional theories. In *Logic Programming and Nonmonotonic Reasoning*, pages 119–131. Springer, 2005.

Paolo Ferraris and Vladimir Lifschitz. Weight constraints as nested expressions. *Theory and Practice of Logic Programming*, 5(1-2):45–74, 2005.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In Veloso [2007], pages 372–379.

Gisèle Fischer-Servi. On modal logic with an intuitionistic base. *Studia Logica*, 36 (4):141–149, 1976.

Melvin Fitting. A kripke-kleene semantics for logic programs. *The Journal of Logic Programming*, 2(4):295–312, 1985.

Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Veloso [2007], pages 386–392.

Martin Gebser, Max Ostrowski, and Torsten Schaub. Constraint answer set solving. In Patricia M. Hill and David Scott Warren, editors, *ICLP*, volume 5649 of *Lecture Notes in Computer Science*, pages 235–249. Springer, 2009. ISBN 978-3-642-02845-8.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(3):1–238, 2012.

Michael Gelfond. On stratified autoepistemic theories. In *AAAI*, volume 87, pages 207–211, 1987.

Michael Gelfond. Autoepistemic logic and formalization of commonsense reasoning (preliminary report). In *Non-Monotonic Reasoning*, pages 176–186. Springer, 1989.

Michael Gelfond. Strong introspection. In Thomas L. Dean and Kathleen McKeown, editors, *AAAI*, pages 386–391. AAAI Press / The MIT Press, 1991. ISBN 0-262-51059-6.

Michael Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):89–116, 1994.

Michael Gelfond. Answer sets. *Handbook of knowledge representation*, 1:285, 2008.

Michael Gelfond. New semantics for epistemic specifications. In James P. Delgrande and Wolfgang Faber, editors, *Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings*, volume 6645 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2011. ISBN 978-3-642-20894-2.

Michael Gelfond. New definition of epistemic specifications. In *KR Seminar*, 2011, April 28. (talk).

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)*, pages 1070–1080. MIT Press, 1988a. ISBN 0-262-61056-6.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988b.

Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation, 1990.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.

Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus. In R. Boyer, editor, *Papers in Honor of Woody Bledsoe*, pages 167–179. Kluwer Academic Publishers, 1991.

Kurt Gödel. Zum intuitionistischen aussagenkalkül. *Anz. Akad. Wiss. Wien*, 69: 65–66, 1932.

David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

Andreas Herzig, Emiliano Lorini, Frédéric Moisan, and Nicolas Troquard. A dynamic logic of normative systems. In Toby Walsh, editor, *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 228–233,

Barcelona, 2011. IJCAI/AAAI. URL http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html. Erratum at http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html.

Arend Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsber. preuss. Akad. Wiss.*, 42-71:158–169, 1930.

Wiebe van der Hoek and Michael Wooldridge. On the dynamics of delegation, cooperation and control: a logical account. In *Proc. AAMAS'05*, 2005.

Wiebe van der Hoek, Dirk Walther, and Michael Wooldridge. On the logic of cooperation and the transfer of control. *JAIR*, 37:437–477, 2010.

Tsutomu Hosoi. The axiomatization of the intermediate propositional systems S_2 of Gödel. 1966.

George Edward Hughes and Maxwell John Cresswell. *A new introduction to modal logic*. Routledge, 2012.

Ulrich Hustadt and Renate A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. In R. Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference (TABLEAUX 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 67–71. Springer, 2000. ISBN 3-540-67697-X. URL http://www.cs.man.ac.uk/~schmidt/publications/HustadtSchmidt00b.html.

Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *The Journal of Logic Programming*, 35(1):39–78, 1998.

Patrick Thor Kahl. *Refining the semantics for epistemic logic programs*. Phd thesis, Texas Tech University, Department of Computer Science, Lubblock, TX, USA, May 2014.

Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors, editor, *Belief revision*, pages 183–203. Cambridge University Press, 1992. (preliminary version in Allen, J.A., Fikes, R., and Sandewall, E., eds., Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf., pages 387–394. Morgan Kaufmann Publishers, 1991).

Robert Kowalski. History of logic programming. *Logic and Computation, In the History of Logic Series*, 9, 2014. to appear.

Marcus Kracht. On extensions of intermediate logics by strong negation. *Journal of Philosophical Logic*, 27(1):49–73, 1998.

Gerhard Lakemeyer and Hector J. Levesque. Only-knowing meets nonmonotonic modal logic. In Brewka et al. [2012]. ISBN 978-1-57735-560-1.

Hector J. Levesque. All I know: a study in autoepistemic logic. *Artificial intelligence*, 42(2-3):263–309, 1990.

Vladimir Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70(1-2):53–72, 1994.

Vladimir Lifschitz. Foundations of logic programming. *Principles of Knowledge Representation*, 3:69–127, 1996.

Vladimir Lifschitz. Answer set planning. pages 23–37. MIT Press, 1999a. ISBN 0-262-54104-1.

Vladimir Lifschitz. Action languages, answer sets, and planning. In *The Logic Programming Paradigm*, pages 357–373. Springer, 1999b.

Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54, 2002.

Vladimir Lifschitz. Thirteen definitions of a stable model. In Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation*, volume 6300 of *Lecture Notes in Computer Science*, pages 488–503. Springer Verlag, 2010. ISBN 978-3-642-15024-1.

Vladimir Lifschitz and Grigori Schwarz. Extended logic programs as autoepistemic theories. In *LPNMR*, pages 101–114, 1993.

Vladimir Lifschitz and Thomas Y. C. Woo. Answer sets in general nonmonotonic reasoning (preliminary report). In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, October 25-29, 1992.*, pages 603–614. Morgan Kaufmann, 1992. ISBN 1-55860-262-3.

Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4): 369–389, 1999.

Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic (TOCL)*, 2(4):526–541, 2001.

J. W. Lloyd. *Foundations of Logic Programming*. Berlin: Springer-Verlag, 1987.

J. Łukasiewicz. Die logik und das grundlagenproblem. In *Les Entretiens de Zurich sur les fondements et la méthode des sciences mathématiques, 6–9 decembre 1938 Zurich*, pages 82–100. 1941.

Michael J. Maher. A transformation system for deductive database modules with perfect model semantics. *Theoretical Computer Science*, 110(2):377–403, 1993.

Victor W. Marek and Miroslaw Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm*, pages 375–398. Springer, 1999.

John McCarthy. Circumscription, a form of nonmonotonic reasoning. *Artificial Intelligence Journal*, 13(1):27–39, 1980.

John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence Journal*, 28(1):1038–1044, 1986.

Robert C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ, 1985a.

Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1), 1985b.

Robert C. Moore. Autoepistemic logic. 1988.

David Nelson. Constructible falsity. *The Journal of Symbolic Logic*, 14(01):16–26, 1949.

Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4): 241–273, 1999.

David Pearce. A new logical characterisation of stable models and answer sets. In Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming, Proc. NMELP 96*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70. Springer Verlag, 1996. ISBN 3-540-62843-6.

David Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3–41, 2006.

David Pearce and Levan Uridia. An approach to minimal belief via objective belief. In Toby Walsh, editor, *IJCAI*, pages 1045–1050. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.

David Pearce, Inman P. de Guzmán, and Agustín Valverde. A tableau calculus for equilibrium entailment. In Roy Dyckhoff, editor, *TABLEAUX*, volume 1847 of *LNCS*, pages 352–367. Springer Verlag, 2000. ISBN 3-540-67697-X.

Halina Przymusinska and Teodor C. Przymusinski. Weakly perfect model semantics for logic programs. In *ICLP/SLP*, pages 1106–1120, 1988.

Teodor C. Przymusinski. Perfect model semantics. In *ICLP/SLP*, pages 1081–1096, 1988a.

Teodor C. Przymusinski. On the relationship between logic programming and nonmonotonic reasoning. In *AAAI*, pages 444–448, 1988b.

Teodor C Przymusinski. Three-valued formalizations of non-monotonic reasoning and logic programming. In *Proceedings of the first international conference on principles of knowledge representation and reasoning*, pages 341–348. Morgan Kaufmann Publishers Inc., 1989.

Raymond Reiter. *On closed world databases*. Springer, 1978.

Raymond Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13 (1):81–132, 1980.

Renate A. Schmidt and Dmitry Tishkovsky. A general tableau method for deciding description logics, modal logics and related first-order fragments. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning (IJCAR 2008)*, volume 5195 of *Lecture Notes in Computer Science*, pages 194–209. Springer, 2008. ISBN 978-3-540-71069-1. doi: http://dx.doi.org/10.1007/978-3-540-71070-7_17. URL http://www.cs.man.ac.uk/~schmidt/publications/SchmidtTishkovsky08b.html.

Roberto Sebastiani and Armando Tacchella. Sat techniques for modal and description logics. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 781–824. IOS Press, 2009. ISBN 978-1-58603-929-5.

Alex K. Simpson. *The proof theory and semantics of intuitionistic modal logic*. Phd thesis, University of Edinburgh, University of Edinburgh, College of Science and Engineering, School of Informatics, November 1994.

Martin Slota and João Leite. Robust equivalence models for semantic updates of answer-set programs. In Brewka et al. [2012]. ISBN 978-1-57735-560-1.

Martin Slota and João Leite. A unifying perspective on knowledge updates. In Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin, editors, *JELIA*, volume 7519 of *Lecture Notes in Computer Science*, pages 372–384. Springer, 2012b. ISBN 978-3-642-33352-1.

Mirosław Truszczyński. Revisiting epistemic specifications. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, pages 315–333. Springer, 2011.

Wiebe van der Hoek and Michael Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, 2005.

Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3):619–649, 1991.

Manuela M. Veloso, editor. *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007.

N. N. Vorob'ev. A constructive propositional calculus with strong negation. In *Doklady Akademii Nauk SSR*, volume 85, pages 465–468, 1952.

Kewen Wang and Yan Zhang. Nested epistemic logic programs. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005, Diamante, Italy, September 5-8, 2005, Proceedings*, volume 3662 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2005. ISBN 3-540-28538-5.

Yan Zhang. Computational properties of epistemic logic programs. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 308–317. AAAI Press, 2006. ISBN 978-1-57735-271-6.

Yan Zhang and Norman Y. Foo. A unified framework for representing logic program updates. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 707–713. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X.