Synonymous theories and knowledge representations in answer set programming☆

D. Pearcea,*,1, A. Valverdeb,2

a Universidad Politécnica de Madrid, Spain
b Universidad de Málaga, Spain

A R T I C L E   I N F O

Article history:
Received 15 August 2009
Received in revised form 7 July 2010
Accepted 10 February 2011
Available online 15 March 2011

Keywords:
Answer set programming
Equilibrium logic
Synonymous theories
Interpretability theory

A B S T R A C T

Even within a single knowledge representation system there are often many different ways to model a given domain and formalise a reasoning problem specified over the domain. In particular, two knowledge descriptions can be semantically equivalent even if they are expressed in quite different languages or vocabularies. This paper proposes and studies a concept of synonymy that applies to equivalent theories formulated in distinct vocabularies. We suggest a set of general desiderata or criteria of adequacy that any reasonable synonymy concept should satisfy. We then analyse a specific concept of synonymy within answer set programming (ASP), a framework that is currently being applied with success in many areas of knowledge technology. We characterise this concept in different ways, show that it satisfies the prescribed criteria of adequacy, and illustrate how it can be applied to a sample problem arising in knowledge representation and reasoning. As a logical framework we use quantified equilibrium logic based on a first-order version of the logic of here-and-there. This serves as an adequate formal foundation for ASP and allows us to obtain a logical account of the synonymy relation.

1. Introduction

This paper is about how different descriptions of a piece of knowledge may be equivalent even if they are expressed in different vocabularies or signatures. It is a type of problem that arises in many disciplines. In logic and the foundations of mathematics one can formalise the idea that mathematical theories, say from algebra, geometry, number theory or set theory, can be presented in various ways using different primitive concepts. In the philosophy of science one is often interested in how equivalent scientific concepts can arise from apparently different theories, how one and the same theory might be expressed or logically reconstructed in distinct ways, or how, as in wave-particle duality in physics, apparently alternative or even rival descriptions of knowledge might actually be partly or fully equivalent. In software engineering one may be faced with the problem of transforming a piece of code written in one programming language into “equivalent” code written in a completely different language. More generally, compiler design faces this task in a systematic fashion. In highly distributed, multi-agent systems, agreement technologies may be needed to determine when agents are using differ-
ent terms in the same way.\textsuperscript{3} In the area of ontologies there is currently much interest in the problem of matching and merging of ontologies. In general one cannot assume that, even within a single domain, different ontologies will use the same vocabulary, so the merging task needs to isolate parts or sub-ontologies that while differently expressed are actually equivalent. Similar problems arise in other areas of knowledge-based reasoning such as machine learning, AI planning or causal theories of action where it can be important to know whether two knowledge descriptions are essentially different or whether they are merely linguistic variants of basically the same thing. In these areas, several of the example types we encounter arise in practice as a result of changing to a new representation on certain pragmatic grounds like efficiency, while aiming to preserve meaning.

1.1. Focus of the paper

Our work will focus on the area of knowledge technologies and on the equivalence of knowledge descriptions formalised in different vocabularies. Knowledge descriptions will be viewed simply as theories in the usual syntax of first-order logic. The underlying representation framework will be that of answer set programming (ASP), interpreted in a standard way using quantified equilibrium logic \[1,2\]. Since the underlying logic will be fixed throughout, our work does not make contact with studies of embedding relations between different nonmonotonic logics, such as in \[3\]. Our focus is rather on relating different conceptual schemes within one logical framework. While ASP has been developed as a formalism for logic programs viewed as sets of nonmonotonic rules, the basic semantics has now been extended to arbitrary first-order theories and many enrichments of the language of normal and disjunctive programs have been implemented. Although answer set solvers may use different programming dialects and apply a variety of special constructions, like weight constraints, cardinality constraints and aggregates, they share a similar underlying semantics and the special constructions can be interpreted in the full first-order syntax. In this way our approach should be widely applicable across different ASP systems.

1.2. General answer set programs and equilibrium logic

Answer set programming is now a well established approach to knowledge representation and declarative problem solving in many application domains. This is largely thanks to the development of practical and efficient solvers such as DLV \[4\], GnT \[5\], smodels \[6\], Cmodels\textsuperscript{4}, ASSAT\textsuperscript{5}, CLASP\textsuperscript{6}, ASpRiX\textsuperscript{7} and others. AI applications include planning and diagnosis, as exemplified in a prototype decision support system for the space shuttle \[7\], the management of heterogeneous data in information systems, as performed in the INFOMIX project\textsuperscript{8}, the representation of ontologies in the semantic web allowing for default knowledge and inference, as discussed in \[8\], as well as compact and fully declarative representations of hard combinatorial problems such as n-Queens, Hamiltonian paths, Towers of Hanoi, and so on.\textsuperscript{9}

In order to analyse problems of equivalence and synonymy in ASP we shall use Quantified Equilibrium Logic (QEL) as developed in \[10–12\]. QEL serves as an adequate logical foundation for answer set programs with variables but it also permits a straightforward definition of answer set (or equilibrium model) for arbitrary theories in first-order logic. The (monotonic) logical basis for QEL is the non-classical logic of Quantified Here-and-There, QHT (see also \[1\]). By expanding the language to include new predicates, this logic can be embedded in classical first-order logic \[2\], and this permits an alternative but equivalent formulation of the concept of answer set for first-order formulas, expressed in terms of classical, second-order logic \[13\]. The latter definition of answer set has been further studied in \[14,15\] where the basis of a first-order programming language, RASPL-1, is described. An alternative approach to a first-order ASP language is developed in \[16\].

There are several advantages to treating full first-order theories in QEL. First, this logic captures all the usual semantics for different ASP dialects. Using the full first-order syntax it is also adequate for all the standard constructions such as cardinality constraints and aggregates. In addition, we don’t have to regard explicit definitions (the cornerstone of our synonymy concept) as special constructions or as meta-level devices: by having a rich enough syntax they become ordinary object language expressions.

1.3. Some motivating examples

Let us consider some examples arising out of different areas of knowledge representation, including planning, causal action theory, and combinatorial problem solving.

\textsuperscript{3} See the European COST Action IC0801 Agreement Technologies, http://www.agreement-technologies.eu and the Spanish research project of the same name: http://www.agreement-technologies.org.

\textsuperscript{4} http://www.cs.utexas.edu/users/tag/cmodels/.

\textsuperscript{5} http://assat.cs.ust.hk/.

\textsuperscript{6} http://www.cs.uni-potsdam.de/clasp/.

\textsuperscript{7} http://www.info.univ-angers.fr/pub/claire/asperix/.

\textsuperscript{8} http://sv.mat.unical.it/infomix/.

\textsuperscript{9} For these kinds of examples as well as a thorough introduction to ASP, see \[9\]. See also the ASP benchmarks available at http://www.cs.kuleuven.be/~dtai/events/ASP-competition/index.shtml.
1. **Operator splitting.** In certain kinds of planning problems Kautz and Selman [17] showed that it is advantageous to split an $n$-place predicate into several predicates of smaller arity. For instance, instead of using the predicate

$$move(x, y, z, i)$$

to say that $x$ is moved from $y$ to $z$ at time $i$, they use three predicates

$$object(x, i) \land source(y, i) \land destination(z, i).$$

In this case the practical concern is to speed up search. Similar examples of relational splitting occur in areas of machine learning.

2. **Action attribute symbols in causal action theories** [18]. The use of action attributes is very similar to the first case, but now the aim is to improve elaboration tolerance; i.e. to make the theory easier to modify when new objects and new scenarios are introduced. In the famous problem of missionaries and cannibals, instead of $cross(V, L)$ for $V$ a vessel, and $L$ a location, one writes

$$cross_{\text{in}}(V), cross_{\text{to}}(L).$$

In [18], Lifschitz analyses in the Causal Calculator many of the 19 variants of the missionaries and cannibals problem proposed by McCarthy. Using action attribute symbols he is able to show how each new scenario can be dealt with merely by the addition of new postulates.

3. **Dual representations.** Sometimes quite different representations of a problem come about because of a slight change of viewpoint. A relatively common example arises in Knowledge Representation when dealing with puzzle-like scenarios such as chess, 8-queens, $n$-puzzle, etc., where we have a board or an array and a set of possibly different pieces or tiles. In these cases we can choose different representations that are in a sense dual to each other. For instance, considering the 8-puzzle, we can represent each tile $T$ as a “main object” and its row $X$ and column $Y$ in the board as its attributes, so that we could have fluents like $row(T, X)$ and $column(T, Y)$ that vary over time. A dual representation could consider instead each board cell (identified by the pair $(X, Y)$) as the main object, and its content as an attribute, so we would have a fluent like $content(X, Y, T)$. Actions may also have dual representations. For instance, in the same example and assuming that tiles are the main objects, we could decide to shift a tile $T$ that is next to the empty position (the hole) so we have an action $move_{\text{tile}}(T, D)$ in some direction $\epsilon \{\text{up, down, right, left}\}$, or we could see the movement as a shift of the hole itself in the opposite direction with an action $move_{\text{hole}}(D')$, assuming that the hole is unique.

4. **Reification.** In another example involving causal theories of action [19] Balduccini and Gelfond propose an architecture for a software agent that operates a physical device and is capable of monitoring, testing and repairing the device’s components. They build a program to find candidate diagnoses using answer set programming. The problem is to compare two different but apparently equivalent ways to manage the preconditions of a law, one using lists (not available in most ASP implementations) and one without. The problem is described in detail in Section 6 below.

5. **Quantifier elimination by introduction of auxiliary predicates.** The standard ASP languages do not currently allow one to formulate rules containing explicit quantifiers within their bodies. However, as Cabalar [20] and Lee and Palla [21] have recently discussed, the use of existential quantifiers in rule bodies can lead to a more compact and intuitive modelling of a problem. For instance, as Cabalar mentions, in ASP one might formulate a typical default law of inertia arising in knowledge representation by means of these rules:

$$holds(F, V, do(A, S)) \leftarrow holds(F, V, S), \quad \text{not } ab(F, V, A, S), \quad (1)$$

$$ab(F, V, A, S) \leftarrow holds(F, W, do(A, S)), \quad W \neq V \quad (2)$$

where $V, W$ are potential values for a fluent $F$, and $A$ and $S$ range over actions and situations, respectively. The predicate $ab$ is an auxiliary expression used to specify that there is some value of $F$ different from $V$. In other words one could use the following, more succinct formulation:

$$holds(F, V, do(A, S)) \leftarrow holds(F, V, S),$$

$$\text{not } \exists W (\text{holds}(F, W, do(A, S)), W \neq V) \quad (3)$$

removing the need for the predicate $ab$. An ASP semantics for such a use of existential quantification is readily given in equilibrium logic or within the general theory of stable models. Cabalar in [20] describes a general method for replacing these quantifiers by auxiliary predicates leading to rules that can be accepted by current ASP solvers.\[10\] The main problem is to show that the two representations are equivalent.

---

\[10\] A similar technique has been implemented in [21].
1.4. Aim of the paper

Our aim is to capture a concept of synonymy, or equivalence, for theories formulated in different vocabularies or signatures in the language of ASP. The framework we propose should be able to handle the kinds of intertheoretic equivalences arising in the above examples from knowledge representation and declarative problem solving. By using a logical reconstruction in QHT and equilibrium logic, we assume that there is single underlying logical language for ASP. Nevertheless, as we shall see from examples, we can also treat cases where differences of vocabulary arise partly out of differences in the underlying programming syntax of different ASP implementations. There are also many specialised languages built on top of ASP systems, such as the action language A, or the planning language K based on the DLV system. Again, as long as we know how to interpret these higher-level languages into basic answer set semantics, there should be no difficulty in treating problems of synonymy and equivalence for theories within or even across these special languages.

An important property of our concept of synonymy is that it should be robust under theory extensions. This is not a self-evident feature of equivalence concepts in nonmonotonic reasoning, where we do not have the usual replacement theorems. Let us consider the case of two theories formulated in the same vocabulary. They may have the same answer sets yet behave very differently once they are embedded in some larger context. For a robust or modular notion of equivalence one should require that programs behave similarly when extended by any further programs. This leads to the following concept of strong equivalence: programs $\Pi_1$ and $\Pi_2$ are strongly equivalent, in symbols $\Pi_1 \equiv_2 \Pi_2$, if and only if for any $\Sigma$, $\Pi_1 \cup \Sigma$ is equivalent to (has the same answer sets as) $\Pi_2 \cup \Sigma$. This has been recognised as providing an important conceptual and practical tool for program simplification, transformation and optimisation. Following its initial study in [22], the concept of strong equivalence for logic programs in ASP has given rise to a substantial body of further work looking at different characterisations [23,24], new variations and applications of the idea [25–27], as well as developing systems to test for strong equivalence [26,28]. Recently, some of this work on program transformation [29,30] has been extended to the first-order case. While strong equivalence and some related concepts are indeed robust, it is assumed that the programs being compared are formulated in the same vocabulary, or at least that any differences of vocabulary are semantically unimportant.11

As we can see from the kinds of examples mentioned in the previous section, this assumption is quite restrictive and not always realistic. It means that until now no systematic tools have been developed to check whether different representations are really semantically equivalent when they are intended to be.

While our technical characterisations of synonymy are specific to the framework of answer set programming, our general approach should be more widely applicable to other KRR and nonmonotonic formalisms. This is true in particular of the general criteria of adequacy that we propose for a synonymy concept. Even our specific methods, based on the theory of interpretations and Beth’s Theorem, may be re-usable for other formalisms, if their underlying logics permit. These methods are adaptations of techniques already used in classical logic that cover examples from mathematics, as well as empirical theories from the natural or social sciences.

1.5. Outline and main results

We start following [31] by considering formal and informal desiderata that a concept of synonymy should fulfil. In Section 3 we then review quantified equilibrium logic and its relation to answer set semantics. We present the main characterisation of strong equivalence from [1]. In Section 4 we discuss the definability of concepts and interpretability between theories. In Section 5 we turn to our main question, how to define a strong concept of equivalence or synonymy for theories in quantified equilibrium logic. We give different characterisations of this concept and show that it fulfils the adequacy conditions discussed in Section 2. The main characteristics of this concept are as follows. Theories $\Pi_1$ and $\Pi_2$ in distinct languages are said to be synonymous if each is bijectively interpretable in the other. In particular, this means that there is faithful interpretation of each theory in the other and a one-one correspondence between the models of the two theories. This correspondence preserves the property of being an equilibrium model or answer set. In addition, $\Pi_1$ has a definitional extension that is strongly equivalent to a definitional extension of $\Pi_2$. Moreover, in a suitable sense, $\Pi_1$ and $\Pi_2$ remain equivalent or synonymous when extended by the addition of new formulas.

As an illustration of our framework, in Section 6 we treat in detail an example from causal action theory applied to diagnostic problems. Section 7 discusses some further aspects of synonymy, while Section 8 reviews literature and related work. Conclusions are drawn in Section 9.

2. Synonymous theories

What does it mean to say that two theories, $\Pi_1$ and $\Pi_2$, in different languages or signatures, $L_1$ and $L_2$, are synonymous? We consider six desiderata D1–D6 that we believe should be satisfied by any basic concept of synonymy. They are quite general and should be applicable to any theories describing or modelling some knowledge domain; notice that D4 takes account of the special nature of a nonmonotonic knowledge representation and reasoning system.

---

11 The assumption is partly relaxed in the case of projective equivalence, where only a part of the vocabulary is assumed to be common to two theories [30]. But then equivalence is only analysed with respect to this shared component.
D1. Translatability. The language $L_1$ of $\Pi_1$ should be translatable, via a mapping, say $\tau$, into the language $L_2$ of $\Pi_2$. The translation $\tau$ should uniformly reflect the structure of the source language, so we require it to be recursive.

D2. Semantic correspondence. There should be a corresponding correlation between the structures of $L_1$ and $L_2$, in particular a mapping $\Phi$ from $L_2$-structures to $L_1$-structures that respects the translation $\tau$ in the sense that for any $L_2$-structure $I$ and $L_1$-formula $\varphi$,

$$\Phi(I) \models \varphi \iff I \models \tau(\varphi).$$

D3. Equivalence. Given the syntactic correlation between their languages, $\Pi_1$ and $\Pi_2$ should be in a strong sense equivalent.

D4. Intended models. The semantic correlation should respect the intended or preferred models of the two theories. In the present case this means preserving the property of being an equilibrium model or answer set: $M$ is an answer set of $\Pi_2$ iff $\Phi(M)$ is an answer set of $\Pi_1$.

D5. Symmetry. If $\Pi_1$ is synonymous with $\Pi_2$ under the previous mappings, then under corresponding mappings $\Pi_2$ should be synonymous with $\Pi_1$. In general we expect synonymy to be an equivalence relation.

D6. Robustness. $\Pi_1$ and $\Pi_2$ should remain synonymous under the addition of new formulas, in other words for any $\Sigma$, $\Pi_1 \cup \Sigma$ should be synonymous with $\Pi_2 \cup \tau(\Sigma)$, similarly for $\Pi_2 \cup \Pi$.

The first two conditions provide the cornerstone of any formal approach to intertheory relations. Different kinds of relations between theories are obtained by specifying additional conditions that the mappings should satisfy (see e.g. [32–34]). In our case, the two languages use standard first-order syntax in a non-classical underlying logic. So in our framework of theories $I$ and $F$ should be in a strong sense equivalent.

Perhaps somewhat surprisingly we approach the problem of synonymy via an adaptation of the classical theory of interpretations. Briefly we shall say that theories are synonymous if each is faithfully and bijectively interpreted in the other; this is basically the standard approach followed in classical predicate logic, see e.g. [35,36]. We adapt it here to the case of a nonmonotonic system based on a non-classical logic.

3. Review of quantified equilibrium logic and answer set semantics

We assume the reader has some familiarity with the usual definition of stable model or answer set semantics for logic programs [37] as well as the way in which answer sets can be used to encode and solve different kinds of reasoning problems [9]. As a logical framework for ASP we use the logic of here-and-there and its nonmonotonic extension (quantified) equilibrium logic. For the propositional version of the logic HT of here-and-there and an overview of propositional equilibrium logic, see [38]. Usually in quantified equilibrium logic we include a second, strong negation operator as occurs in several ASP dialects. In this paper we shall restrict attention to the language with a single negation symbol, ‘¬’. In particular, we shall work with a quantified version of the logic HT, with the usual logical constants ‘¬’, ‘∧’, ‘∨’, ‘→’, quantifiers ‘∀’, ‘∃’, and the defined constant ‘↔’. In other respects we follow the treatment of [12].

For the remainder of the paper we consider languages $\mathcal{L} = \{C, F, P\}$, built over a set of constants, $C$, a set of functions, $F$, and a set of predicates, $P$; the three sets of symbols are disjoint and each predicate symbol and each function symbol has an assigned arity. Atoms and formulas are constructed as usual; closed formulas, or sentences, are those where no variable appears outside the scope of a quantifier. A theory is a set of sentences. Variable-free terms, atoms, formulas, or theories are also called ground.

We regard structures as sets of atoms built over arbitrary non-empty domains, $D$; we denote by $\text{At}(D, F, P)$ the set of atomic sentences of $(D, F, P)$ (if $D = C$, we obtain the set of atomic sentence of the language $\mathcal{L} = \{C, F, P\}$); and we denote by $T(D, F)$ the set of ground terms of $(D, F, P)$. If $\mathcal{L} = \{C, F, P\}$ and $\mathcal{L}' = \{C', F', P'\}$, we write $\mathcal{L} \subseteq \mathcal{L}'$ if $C \subseteq C'$, $F \subseteq F'$ and $P \subseteq P'$.

By an $\mathcal{L}$-interpretation over a set $D$ we mean a subset of $\text{At}(D, F, P)$. A classical $\mathcal{L}$-structure can be regarded as a tuple $I = \langle (D, I), I^* \rangle$ where $I^*$ is an $\mathcal{L}$-interpretation over $D$ and $I : T(C \cup D, F) \rightarrow D$, called the assignment, verifies that $I(d) = d$ for all $d \in D$ and is recursively defined. If $D = T(C, F)$ and $I = \text{id}$, $I$ is known as a Herbrand structure. On the other hand, a here-and-there $\mathcal{L}$-structure with static domains, or $\text{QHT}^\mathcal{L}(\mathcal{L})$-structure, is a tuple $I = \langle (D, I), I^h, I^f \rangle$ where $\langle (D, I), I^h \rangle$ and $\langle (D, I), I^f \rangle$ are classical $\mathcal{L}$-structures such that $I^h \subseteq I^f$.

---

12 We can think of the objects in $D$ as additional constants; this approach allow us to use a simplified notation where the objects are not distinguished from their names.

13 That is, for every $a \in C$, $I(a) \in D$ and for every $f \in F$ with arity $n$, a mapping $f^I : D^n \rightarrow D$ is defined; so the recursive definition is given by $I(f(t_1, \ldots, t_n)) = f^I(I(t_1), \ldots, I(t_n))$. 
If $I = (D, I^h, I^t)$ is an $L'$-structure and $L' \supseteq L$, we denote by $I_{\mid L}$ the restriction of $I$ to the sublanguage $L$:

$I_{\mid L} = (D, I_{\mid L}, I^h_{\mid L}, I^t_{\mid L})$

where: $I_{\mid L}$ is the restriction of the assignment $I$ to the set of terms built with $D$ and the constants and functions from $L$; and, for $w \in \{h, t\}$, $I^w_{\mid L}$ is obtained by removing from $I^w$ any atom with predicates, constants or functions in $L'$ which are not in $L$.

Thus we can think of a here-and-there structure $I$ as similar to a first-order classical model, but having two parts, or components, $h$ and $t$ that correspond to two different points or “worlds”, ‘here’ and ‘there’, in the sense of Kripke semantics for intuitionistic logic [39], where the worlds are ordered by $h < t$. At each world $w \in \{h, t\}$ one verifies a set of atoms $I^w$ in the expanded language for the domain $D$. We call the model static, since, in contrast, to say, intuitionistic logic, the same domain serves each of the worlds. Since $h < t$, whatever is verified at $h$ remains true at $t$. The satisfaction relation for $I$ is defined so as to reflect the two different components, so we write $I, w \models \varphi$ to denote that $\varphi$ is true in $I$ with respect to the $w$ component. Although we only need to define the satisfaction relation in $L = \langle C, F, P \rangle$, the recursive definition forces us to consider formulas from $(C \cup D, F, P)$. In particular, if $p(t_1, \ldots, t_n) \in \text{At}(C \cup D, F, P)$ then $I, w \models p(t_1, \ldots, t_n)$ iff $p(I(t_1), \ldots, I(t_n)) \in I^w$ for every $t_1, \ldots, t_n \in T(C \cup D, F)$. Then $\models$ is extended recursively as follows14:

- $I, w \models \varphi \land \psi$ if $I, w \models \varphi$ and $I, w \models \psi$.
- $I, w \models \varphi \lor \psi$ if $I, w \models \varphi$ or $I, w \models \psi$.
- $I, t \models \varphi \rightarrow \psi$ if $I, t \not\models \varphi$ or $I, t \models \psi$.
- $I, h \models \varphi \rightarrow \psi$ if $I, t \models \varphi \rightarrow \psi$ and $I, h \not\models \varphi$ or $I, h \models \psi$.
- $I, w \models \neg \varphi$ if $I, t \not\models \varphi$.
- $I, t \models \forall x \varphi(x)$ if $I, t \models \varphi(d)$ for all $d \in D$.
- $I, h \models \forall x \varphi(x)$ if $I, t \models \forall x \varphi(x)$ and $I, h \models \varphi(d)$ for all $d \in D$.
- $I, w \models \exists x \varphi(x)$ if $I, w \models \varphi(d)$ for some $d \in D$.

Truth of a sentence in a model is defined as follows: $I \models \varphi$ iff $I, w \models \varphi$ for each $w \in \{h, t\}$. A sentence $\varphi$ is valid if it is true in all models, denoted by $\models \varphi$. A sentence $\varphi$ is a consequence of a set of sentences $\Pi$, denoted $\Pi \models \varphi$, if every model of $\Pi$ is a model of $\varphi$.

The resulting logic is called Quantified Here-and-There Logic with static domains, and denoted in [1] by $QHT^s$. In terms of satisfiability and validity this logic is equivalent to the logic introduced in [11].

A complete axiomatisation of $QHT^s$ can be obtained as follows [1]. We take the axioms and rules of first-order intuitionistic logic [39] and add the axiom of Hosoi [40]:

$$\alpha \lor (\neg \beta \lor (\alpha \rightarrow \beta))$$

which determines 2-element here-and-there models in the propositional case, together with the axiom:

$$\exists x (\alpha(x) \rightarrow \forall x \alpha(x))$$

We also consider the equality predicate, $\equiv \not\equiv P$, interpreted by the following condition for every $w \in \{h, t\}$.

- $I, w \models t_1 \equiv t_2$ iff $I(t_1) = I(t_2)$.

To obtain a complete axiomatisation, we need to add the axiom of “decidable equality”

$$\forall x \forall y (x \equiv y \lor x \not\equiv y).$$

We denote the resulting logic by $QHT^s_{\equiv}(L)$ and its inference relation by $\models \equiv$. More details can be found in [1] where in particular a strong completeness theorem is proved, that is: for any theory $\Gamma$ and any formula $\varphi$, $\Gamma \models \varphi$ if and only if $\Gamma \models \varphi$.

As usual in first order logic, satisfiability and validity are independent of the signature.

**Proposition 1.** Suppose that $L' \supseteq L$, $\Pi$ is a theory in $L$ and $M$ is an $L'$-model of $\Pi$. Then $M_{\mid L}$ is an $L'$-model of $\Pi$.

**Proposition 2.** Suppose that $L' \supseteq L$ and $\varphi \in L$. Then $\varphi$ is valid (resp. satisfiable) in $QHT^s_{\equiv}(L)$ if and only if it is valid (resp. satisfiable) in $QHT^s_{\equiv}(L')$.

This proposition allows us to omit reference to the signature in the logic so it can be denoted simply by $QHT^s_{\equiv}$.

---

14 The following corresponds to the usual Kripke semantics for intuitionistic logic given our assumptions about the two worlds $h$ and $t$ and the single domain $D$; see e.g. [39].
In the context of logic programs, the following assumptions often play a role. In the case of both classical and $\text{QHT}^\text{u}_s$ models, we say that the parameter names assumption (PNA) applies in case $I|_{T(C,F)}$ is surjective, i.e., there are no unnamed individuals in $D$; the unique names assumption (UNA) applies in case $I|_{T(C,F)}$ is injective; in case both the PNA and UNA apply, the standard names assumption (SNA) applies, i.e. $I|_{T(C,F)}$ is a bijection.

3.1. Equilibrium models

As in the propositional case, quantified equilibrium logic is based on a suitable notion of minimal model.

**Definition 1.** Among quantified here-and-there structures we define the order $\subseteq$ as follows: $(D, I, l^b, l^f) \subseteq (D', J, j^b, j^f)$ if $D = D'$, $I = J$, $l^b = j^b$ and $l^f \subseteq j^f$. If the subset relation holds strictly, we write `$\subset$'.

**Definition 2 (Equilibrium model).** Let $\Pi$ be a theory and $I = ((D, I), l^b, l^f)$ a model of $\Pi$.

1. $I$ is said to be total if $l^b = l^f$.
2. $I$ is said to be an equilibrium model of $\Pi$ (or short, we say: “$I$ is in equilibrium”) if it is minimal under $\subseteq$ among models of $\Pi$, and it is total. It is denoted by $I \vdash \Pi$.

Notice that a total here-and-there model of a theory $\Pi$ is equivalent to a classical first order model of $\Pi$.

The logic defined by the equilibrium models is called Quantified Equilibrium Logic (QEL) and it is also independent of the language, as seen by the following result.

**Proposition 3.** Let $\Pi$ be a theory in $\mathcal{L}$ and $\mathcal{M}$ an equilibrium model of $\Pi$ in $\text{QHT}^\text{u}_s(\mathcal{C}')$ with $\mathcal{L}' \supset \mathcal{L}$. Then $\mathcal{M}|_{\mathcal{L}}$ is an equilibrium model of $\Pi$ in $\text{QHT}^\text{u}_s(\mathcal{L})$.

3.1.1. Relation to other definitions of answer set

We assume the reader is familiar with the usual definitions of answer set based on Herbrand models and ground programs, e.g. [9]. Two variations of this semantics, the open [16] and generalised open answer set [41] semantics, consider non-ground programs and open domains, thereby relaxing the PNA.

For the present version of QEL the correspondence to answer sets can be summarised as follows (see [11,12,42]). If $\varphi$ is a universal sentence in $\mathcal{L} = (C,F,P)$,\(^15\) a total $\text{QHT}^\text{u}_s$ model $((D, I), T, T)$ of $\varphi$ is an equilibrium model of $\varphi$ iff $(T, T)$ is a propositional equilibrium model of the grounding of $\varphi$ with respect to the universe $U$.

By the usual convention, when $\Pi$ is a logic program with variables we consider the models of its universal closure expressed as a set of logical formulas. It follows that if $\Pi$ is a logic program (of any form), a total $\text{QHT}^\text{u}_s$ model $((D, I), T, T)$ of $\Pi$ is an equilibrium model of $\Pi$ iff it is a generalised open answer set of $\Pi$ in the sense of [41]. If we assume all models are UNA-models, we obtain the version of QEL found in [11]. There, the following relation of QEL to (ordinary) answer sets for logic programs with variables was established. If $\Pi$ is a logic program, a total UNA-$\text{QHT}^\text{u}_s$ model $((D, I), T, T)$ of $\Pi$ is an equilibrium model of $\Pi$ iff it is an open answer set of $\Pi$.

In [13] a new definition of stable model for arbitrary first-order formulas is provided, defining the property of being a stable model syntactically via a second-order condition. However [13] also shows that the new notion of stable model is equivalent to that of equilibrium model defined here. In a sequel to this paper, [14] applies the new definition and makes the following refinements. The stable models of a formula are defined as in [13] while the answer sets of a formula are those Herbrand models of the formula that are stable in the sense of [13]. Using this new terminology, it follows that in general stable models and equilibrium models coincide, while answer sets are equivalent to SNA-$\text{QHT}^\text{u}_s$ models that are equilibrium models.

Since we capture a general notion of answer set for arbitrary first-order theories, our framework of $\text{QHT}^\text{u}_s$ and QEL is widely applicable to knowledge representation problems reconstructed in different ASP languages. One area where there is less uniformity is the treatment of functions in ASP. This is still an active area of theoretical research and there are different approaches, e.g. [43–46]. The introduction of function symbols in ASP systems is still at an early stage and most current solvers are not yet equipped with them. As we have seen, our approach here uses a rather classical notion of total function with decidable equality. The techniques used below could however be adapted, if necessary, to other treatments of functions in ASP.

3.2. Strong equivalence for theories

Before turning to the concept of synonymy in ASP, we review briefly the notion of strong equivalence that captures a robust form of equivalence for theories expressed in the same language.

\(^{15}\) I.e. a sentence in prenex form all of whose quantifiers are universal; see [11] for prenex forms in $\text{QHT}^\text{u}_s$.\]
The study of strong equivalence for logic programs and nonmonotonic theories was initiated in [22]. It has since become an important tool in ASP as a basis for program transformation and optimisation. We say that two sets $\Pi_1$, $\Pi_2$ of first-order sentences are strongly equivalent if for every set $\Sigma$ of first-order sentences, possibly of a larger signature, the sets $\Pi_1 \cup \Sigma$, $\Pi_2 \cup \Sigma$ have the same equilibrium models. Under this definition we have:

**Theorem 1.** (See [1,12].) Two (first-order) theories $\Pi_1$ and $\Pi_2$ are strongly equivalent if and only if they have the same $\text{QHT}_{\Delta_1}$-models.

By strong completeness, it follows that two theories $\Gamma$ and $\Delta$ are strongly equivalent if and only if they are logically equivalent in $\text{QHT}_{\Delta_1}$.

The following results will be useful later when the concepts of definability and interpretation are introduced. In them, we may assume that the formulas are in a suitable normal form, where functions only appear in atoms with equality and without nesting. Here and in the remainder of the paper we use the following notation and terminology. Boldface we may assume that the formulas are in a suitable normal form, where functions only appear in atoms with equality and without nesting.

**Proposition 4.** For every formula $\varphi$, it is possible to build a formula $\psi$, such that $\varphi \equiv \psi$, and the atoms of $\psi$ are of one of the following types:

- $x \equiv a$ for some $a \in C$,
- $f(x_1, \ldots, x_n) \equiv y$ for some $f \in F$ (where every $x_i$ and $y$ are variables),
- $p(x_1, \ldots, x_n)$ (where each $x_i$ is a variable).

**Proof.** (Idea.) The formula $\psi$ is constructed by successive applications of the equivalence in Lemma 1 to replace every atom by an equivalent formula, starting from the innermost function or constant symbol in $\varphi$. \(\square\)

The following example is taken from Enderton [47] and illustrates how this method works in a specific case:

$$\forall x(p(a) \rightarrow q(f(g(x)))) \equiv \forall x(\forall y(y \equiv a \rightarrow p(y)) \rightarrow q(f(g(x))))$$

$$\equiv \forall x(\forall y(y \equiv a \rightarrow p(y)) \rightarrow (g(x) \equiv y \rightarrow q(f(y))))$$

$$\equiv \forall x(\forall y(y \equiv a \rightarrow p(y)) \rightarrow (g(x) \equiv y \rightarrow (\forall z(f(y) \equiv z \rightarrow q(z)))).$$

**4. Definability and interpretability**

Our approach to synonymy uses the theory of definability and interpretations (between theories). In many respects it is close to the classical theory found in logic textbooks, e.g. [47]. However, we need to adapt this theory to our underlying non-classical logic and draw on metalogical properties of $\text{QHT}_{\Delta_1}$. We start with some elements of definability theory.

**Definition 3 (Explicit definability).** Let $\mathcal{L} = \langle C, F, P \rangle$ be a first-order language.

- Let $p \notin P$ be a new predicate symbol and $\Pi$ a theory in $\mathcal{L}' = \langle C, F, P \cup \{p\} \rangle$. The symbol $p$ is said to be explicitly definable in $\Pi$, if there is an $\mathcal{L}$-formula $\delta_p(x)$ such that
  $$\Pi \models \forall x(p(x) \iff \delta_p(x)).$$

$\delta_p$ is called the Defn of $p$.

- Let $f \notin F$ be a new function symbol and $\Pi$ a theory in $\mathcal{L}' = \langle C, F \cup \{f\}, P \rangle$. The symbol $f$ is said to be explicitly definable in $\Pi$, if there is an $\mathcal{L}$-formula $\delta_f(x, y)$ such that
  $$\Pi \models \forall x \forall y(f(x) \equiv y \iff \delta_f(x, y)).$$

$\delta_f$ is called the Defn of $f$.
Let $a \notin C$ be a new constant symbol and $\Pi$ a theory in $L' = (C \cup \{a\} F, P)$. The symbol $a$ is said to be \textit{explicitly definable} in $\Pi$, if there is an $L$-formula $\delta_a(x)$ such that

$$\Pi \models \forall x (a \equiv x \leftrightarrow \delta_a(x)).$$

$\delta_a$ is called the \textit{Defn} of $a$.

\textbf{Definition 4 (Implicit definability).} Let $L'$ be a language obtained from $L$ by adding just one new predicate, function or constant symbol, and let $\Pi$ be a theory in $L'$. The new symbol is said to be \textit{implicitly definable} in $\Pi$ if for any models $M_1$ and $M_2$ of $\Pi$ such that $M_1|_{L} = M_2|_{L}$ we have $M_1 = M_2$.

The strong completeness theorem for $QHT^c_L$ and Proposition 4 allows us to conclude the following characterisation of the implicit definability:

- A predicate $p$ is \textit{implicitly definable} in $\Pi$ iff
  $$\Pi \cup \Pi[p/q] \models \forall x (p(x) \leftrightarrow q(x))$$
  where $q \notin P$ is a new predicate symbol with the same arity as $p$ and $\Pi[p/q]$ is the theory obtained by replacing every occurrence of $p$ by $q$.
- Assuming that the formulas in $\Pi$ are in the form described in Proposition 4, the function $f$ is \textit{implicitly definable} in $\Pi$ iff
  $$\Pi \cup \Pi' \cup \forall x y (q(x, y) \lor \neg q(x, y)) \models \forall x (f(x) \equiv y \leftrightarrow q(x, y))$$
  where $q \notin P$ is a new predicate symbol with the same arity as $f$ plus one, and $\Pi'$ is built from $\Pi$ by replacing every atom $f(x) \equiv y$ by $q(x, y)$.

In other words, $p$ is implicitly definable if whenever the interpretation of the $L$ predicates in models of $\Pi$ is fixed, the interpretation of $p$ becomes fixed also; similarly for functions and constants. The above definitions are readily extended to the case where several new predicate, function and constant symbols are definable in a theory.

When the conditions for explicit and implicit definability are always equivalent, the logic in question is said to have the \textit{Beth property} [48].

\textbf{Proposition 5.} The logic $QHT^c_L$ possesses the Beth property.

\textbf{Proof.} The Beth property is closely related to the property of \textit{interpolation}.\(^{16}\) It can be shown that the interpolation property implies the Beth property in all superintuitionistic predicate logics [48]. Moreover, Ono [49] showed that interpolation holds in the logic $QHT^c$ of quantified here-and-there with constant domains.\(^{17}\) Consequently, $QHT^c$ also has the Beth property.

Lastly, Makimova showed in [50,51] that adding pure equality axioms, e.g. decidable equality axiom, to any superintuitionistic logic preserves the interpolation and Beth properties (see also [48]). Therefore from known results we can conclude that \textit{QHT}$^c$ also possesses the Beth property. However, since these results are generally stated in the literature for function-free languages, we need to adapt them to our present situation, as follows.

Let $\Pi$ be a theory in $L' = (C, F \cup \{f\}, P)$. By Lemma 1, we can assume w.l.o.g. that the occurrences of $f$ in $\Pi$ are always in atoms $f(t) \equiv y$. Let $p_f \notin P$ a new predicate symbol and $\Pi'$ the theory in $L'' = (C, F, P \cup \{p_f\})$ obtained by replacing the atoms $f(t) \equiv y$ by $p_f(t, y)$ and

$$\Pi'' = \Pi' \cup \{\exists x \forall y \forall z (p_f(x, z) \leftrightarrow z \equiv y)\}.$$ 

Trivially, there exists a bijection, $\Phi$, between the models of $\Pi''$ and the models of $\Pi$, which is defined using the following relation:

$$p_f(d_1, \ldots, d_n, d_{n+1}) \in I^h \text{ iff } I(f(d_1, \ldots, d_n)) = d_{n+1}.$$ 

To verify the Beth property, let us assume that $f$ is explicitly definable in $\Pi$, then

$$\Pi \models \forall x \forall y (f(x) \equiv y \leftrightarrow \delta_f(x, y))$$

\(^{16}\) A logic is said to have the interpolation property if whenever $\vdash \varphi \rightarrow \psi$ there exists a sentence $\xi$ (the \textit{interpolant}) such that $\vdash \varphi \rightarrow \xi$ and $\vdash \xi \rightarrow \psi$ where all predicate and constant symbols of $\xi$ are contained in both $\varphi$ and $\psi$.

\(^{17}\) Ono’s axiomatisation of $QHT^c$ uses the constant domains axiom $\forall x (a(x) \lor \beta) \rightarrow (\forall x (a(x) \lor \beta))$, as well as alternative axioms for propositional here-and-there, viz. $p \lor (p \lor (q \lor \neg q))$ and $(p \lor (q \lor (q \lor p)) \lor (p \lor \neg q))$. However, the axioms given here are equivalent to Ono’s.
and thus
\[ \Pi'' \models \forall x \forall y (p_f(x, y) \leftrightarrow \delta_f(x, y)). \]
That is, \( \delta_f \) is a definition of \( p_f \) and, by the Beth property already established for predicate symbols, \( p_f \) is implicitly definable. Let \( M_1 \) and \( M_2 \) two models of \( \Pi \) such that \( M_1 \models M_2 \); then \( \Phi(M_1) \models \Phi(M_2) \); by the Beth property, \( \Phi(M_1) = \Phi(M_2) \) and thus \( M_1 = M_2 \).

The other direction of the Beth property is proved analogously. \( \square \)

4.1. Interpretations between theories

**Definition 5 (Interpretation between languages).** Let \( L_1 = (C_1, F_1, P_1) \) and \( L_2 = (C_2, F_2, P_2) \) be disjoint languages.\(^{18}\) By an interpretation \( \tau \) of \( L_1 \) in \( L_2 \)

1. for each predicate \( p \in P_2 \), an \( L_2 \)-formula \( \delta^p_p \) such that the number of free variables in \( \delta^p_p \) is at most the arity of \( p \);
2. for each constant \( a \in C_1 \), an \( L_2 \)-formula \( \delta^a_a \);
3. for each function \( f \in F_1 \), an \( L_2 \)-formula \( \delta^f_f \) such that the number of free variables in \( \delta^f_f \) is at most the arity of \( p \) plus one.

An interpretation \( \tau \), induces a mapping, also denoted by \( \tau \), from \( L_1 \)-formulas to \( L_2 \)-formulas: first, a formula is converted in the form described in Proposition 4 and then

1. \( \tau(x = y) = x = y \),
2. \( \tau(f(x) = y) = \delta^f_f(x, y) \) for every function \( f \),
3. \( \tau(a = x) = \delta^a_a(x) \), for every constant \( a \),
4. \( \tau(p(x)) = \delta^p_p(x) \),
5. \( \tau \) is extended recursively by \( \tau(\varphi \land \psi) = \tau(\varphi) \land \tau(\psi) \), \( \tau(\varphi \lor \psi) = \tau(\varphi) \lor \tau(\psi) \), \( \tau(\varphi \rightarrow \psi) = \tau(\varphi) \rightarrow \tau(\psi) \), \( \tau(\neg \varphi) = \neg \tau(\varphi) \), \( \tau(\forall x \varphi) = \forall \tau(x) \) and \( \tau(\exists x \varphi) = \exists \tau(x) \).

**Definition 6 (Induced definitions).** For an interpretation \( \tau \), we define the set of definitions induced by \( \tau \) as:
\[ \mathcal{T} = \{ \forall x (p(x) \leftrightarrow \delta^p_p(x)), \forall y (a = y \leftrightarrow \delta^a_a(y)), \forall x \forall y (f(x) = y \leftrightarrow \delta^f_f(x, y)) \} \]

**Definition 7 (Interpretation into a theory).** If \( \tau \) is an interpretation of \( L_1 \) in \( L_2 \) and \( \Pi_2 \) is a theory in \( L_2 \), \( \tau \) is said to be an interpretation of \( L_1 \) in \( \Pi_2 \) if the following conditions hold:

1. for every constant \( a \in C_1 \),
\[ \Pi_2 \models \exists x \forall y (\delta^a_a(y) \leftrightarrow y = x); \]
2. for every function \( f \in F_1 \),
\[ \Pi_2 \models \forall x \exists y \forall z (\delta^f_f(x, z) \leftrightarrow z = y). \]

Any interpretation \( \tau \) of \( L_1 \) in \( \Pi_2 \) induces a mapping \( \Phi_\tau \) from the set of models of \( \Pi_2 \) to the set of \( L_1 \)-structures: if \( I = (\langle D, I \rangle, I^0, I^1) \), then \( \Phi_\tau(I) = (\langle D, J \rangle, J^0, J^1) \) is defined as follows:

- for every \( a \in C_1 \), \( J(a) = d \) if \( I \models \delta^a_a(d) \);
- for every \( f \in F_1 \), \( f^1(d_1, \ldots, d_n) = d_{n+1} \) if \( I \models \delta^f_f(d_1, \ldots, d_n, d_{n+1}) \);
- \( h(t) \in J^w \) iff \( I, w \models \delta^f_f(t) \), for every \( w \in \{h, t\} \).

**Lemma 2.** The mapping \( \Phi_\tau \) is well defined.

**Proof.** The condition (4) in the definition of interpretation guarantees that \( J \) is well defined over the set of constants and condition (5) guarantees that \( f^1 \) is well defined for every \( f \in F_1 \). \( \square \)

\(^{18}\) Any languages can be made disjoint by renaming. Alternatively we can allow that \( L_1 \) and \( L_2 \) have a common sublanguage which any translations simply leave untouched, i.e. the sublanguage is always translated by the identity map.
On the other hand, it is easy to check by induction that for any $\mathcal{L}_1$-sentence $\varphi$, any $w \in \{h, t\}$ and any model $\mathcal{I}$ of $\Pi_2$:

$$\Phi_\tau(\mathcal{I}), w \models \varphi \iff \mathcal{I}, w \models \tau(\varphi)$$

(6)

and therefore

$$\Phi_\tau(\mathcal{I}) \models \varphi \iff \mathcal{I} \models \tau(\varphi).$$

(7)

**Definition 8 (Interpretation between theories).** Let $\mathcal{L}_1 = (C_1, F_1, P_1)$ and $\mathcal{L}_2 = (C_2, F_2, P_2)$ be disjoint languages and $\Pi_1$ and $\Pi_2$ two theories in $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. An interpretation $\tau$ of $\mathcal{L}_1$ in $\Pi_2$ is said to be an interpretation of $\Pi_1$ in $\Pi_2$ if

$$\Pi_1 \models \varphi \implies \Pi_2 \models \tau(\varphi).$$

(8)

For interpretations between theories, the mapping $\Phi_\tau$ maps models of $\Pi_2$ to models of $\Pi_1$.

**Proposition 6.** Let $\tau$ be an interpretation of $\mathcal{L}_1$ in $\Pi_2$, and let $\Phi_\tau$ be the induced map defined above. Then $\tau$ is an interpretation of $\Pi_1$ in $\Pi_2$ if and only if:

$$\mathcal{I} \models \Pi_2 \implies \Phi_\tau(\mathcal{I}) \models \Pi_1.$$  

(9)

**Proof.** ($\implies$) Let us assume that $\tau$ is an interpretation of $\Pi_1$ in $\Pi_2$ and $\mathcal{I}$ a model of $\Pi_2$. If $\varphi \in \Pi_1$, then $\Pi_1 \models \varphi$ and, by (8), $\Pi_2 \models \tau(\varphi)$; so, $\mathcal{I}$ is a model of $\tau(\varphi)$ and, by (7), $\Phi_\tau(\mathcal{I}) \models \varphi$. Therefore $\Phi_\tau(\mathcal{I}) \models \Pi_1$.

($\impliedby$) Let us assume (9), $\Pi_1 \models \varphi$ and let $\mathcal{I}$ be a model of $\Pi_2$. Then, by (9), $\Phi_\tau(\mathcal{I})$ is a model of $\Pi_1$ and thus it is a model of $\varphi$; therefore, by (7), $\mathcal{I}$ is a model of $\tau(\varphi)$ and $\Pi_2 \models \tau(\varphi)$.

**Corollary 1.** An interpretation $\tau$ of $\mathcal{L}_1$ in $\Pi_2$ is an interpretation of $\Pi_1$ in $\Pi_2$ if and only if:

$$\text{If } \mathcal{I} \models \Pi_2, \text{ then } \mathcal{I} \models \tau(\varphi) \text{ for all } \varphi \in \Pi_1.$$  

Generally speaking the map $\Phi_\tau$ associated with an interpretation $\tau$ of $\mathcal{L}_1$ in an $\mathcal{L}_2$-theory does not preserve the ordering $\leq$ between $\mathcal{L}_2$-structures. However the following properties are easy to check and will be useful later:

**Lemma 3.** Let $\tau$ be an interpretation of $\mathcal{L}_1$ in $\Pi_2$, and let $\Phi_\tau$ be the induced map between structures. Let $\mathcal{I}$ be a total model of $\Pi_2$. Then

(i) $\Phi_\tau(\mathcal{I})$ is a total $\mathcal{L}_1$-structure; and

(ii) if $\mathcal{I} \models \Pi_2$ and $\mathcal{I} \not\leq \mathcal{I}$, then $\Phi_\tau(\mathcal{I}) \not\leq \Phi_\tau(\mathcal{I})$.

**Proof.** (i) follows straightforwardly from (6): if $\Phi_\tau(\mathcal{I})$ is a non-total $\mathcal{L}_1$-structure, then for some sentence $\varphi$, we have $\Phi_\tau(\mathcal{I}), t \models \varphi$ and $\Phi_\tau(\mathcal{I}), h \not\models \varphi$. By (6) therefore $\mathcal{I}, t \models \tau(\varphi)$ while $\mathcal{I}, h \not\models \tau(\varphi)$, contradicting the totality of $\mathcal{I}$.

To prove (ii) let us consider a total structure $\mathcal{I}$ and $\mathcal{I}' \not\leq \mathcal{I}$; then using (6) and the definition of $\leq$ we have that for any $\mathcal{L}_1$-formula $\varphi$,

$$\Phi_\tau(\mathcal{I}'), t \models \varphi \iff \mathcal{I}', t \models \tau(\varphi) \iff \mathcal{I}, t \models \tau(\varphi) \iff \Phi_\tau(\mathcal{I}), t \models \varphi$$

and so $\Phi_\tau(\mathcal{I}')$ and $\Phi_\tau(\mathcal{I})$ agree at their $t$ points. On the other hand, by

$$\Phi_\tau(\mathcal{I}'), h \models \varphi \iff \mathcal{I}', h \models \tau(\varphi) \implies \mathcal{I}, t \models \tau(\varphi) \iff \Phi_\tau(\mathcal{I}), t \models \varphi$$

we obtain $\Phi_\tau(\mathcal{I}'), h \models \varphi \implies \Phi_\tau(\mathcal{I}), t \models \varphi$ and therefore $\Phi_\tau(\mathcal{I}') \not\leq \Phi_\tau(\mathcal{I})$, since $\mathcal{I}$ is total.

**Definition 9 (Faithful interpretation).** An interpretation $\tau$ of $\Pi_1$ in $\Pi_2$ is said to be faithful if the converse of (8) also holds, i.e. we have

$$\Pi_1 \models \varphi \iff \Pi_2 \models \tau(\varphi).$$

(10)

As in classical interpretability theory, further special cases of interpretation can be obtained by imposing additional conditions on the syntactic and semantic translations.
Proposition 7. Let $\tau$ be an interpretation of $\Pi_1$ in $\Pi_2$. Then the following are equivalent.

(i) For every $L_2$-formula $\psi(x)$ there is an $L_1$-formula $\varphi(x)$ such that $\Pi_2 \models \forall x(\psi(x) \leftrightarrow \tau(\varphi(x)))$; i.e. $\tau$ is surjective.
(ii) There is an interpretation $\sigma$ of $L_2$ in $L_1$ such that for every $L_2$-formula $\psi(x)$, $\Pi_2 \models \forall x(\psi(x) \leftrightarrow \tau(\sigma(\psi(x))))$; $\sigma$ is called the inverse of $\tau$.
(iii) The mapping $\Phi_\tau$ from models of $\Pi_2$ into models of $\Pi_1$ is an injection.

Proof. (ii) implies (i) is straightforward.

(i) implies (iii) is also easy: let $I_1$ and $I_2$ be two models of $\Pi_2$ such that $\Phi_\tau(I_1) = \Phi_\tau(I_2)$; let $p(t)$ be a basic atom in $L_2$ and $\varphi(t)$ an $L_1$-formula such that $\Pi_2 \models p(t) \leftrightarrow \tau(\varphi(t))$.

$$I_1, w \models p(t) \iff I_1, w \models \tau(\varphi(t)) \quad \text{(because $I_1 \models p(t) \leftrightarrow \tau(\varphi(t))$)}$$

$$\iff \Phi_\tau(I_1), w \models \varphi(t)$$

$$\iff \Phi_\tau(I_2), w \models \varphi(t) \quad \text{(because $\Phi_\tau(I_1) = \Phi_\tau(I_2)$)}$$

$$\iff I_2, w \models \tau(\varphi(t))$$

$$\iff I_2, w \models p(t) \quad \text{(because $I_2 \models p(t) \leftrightarrow \tau(\varphi(t))$).}$$

All of this is valid even if $p(t) = f(s) \equiv a$, and thus we can conclude that $f^{i_1} = f^{i_2}$ for every $f$. Therefore $I_1 = I_2$ and $\Phi_\tau$ is injective.

To show that (iii) implies (ii), one applies the Beth property. First, we shall show that in $L_2 \cup L_1$ the theory $\Pi = \Pi_2 \cup \tau$ implicitly defines the atoms of $L_2$: let $M_1$ and $M_2$ be two models of $\Pi$ such that $M_1|_{L_1} = M_2|_{L_1}$; to prove that $M_1 = M_2$, we only need to prove that $M_1|_{L_2} = M_2|_{L_2}$:

- By Proposition 1, the restrictions $M_1|_{L_2}$ and $M_2|_{L_2}$ are models of $\Pi_2$.
- For every atomic formula $p(x)$, $M_1 \models \forall x(p(x) \leftrightarrow \delta_0^\Pi(x))$: so, for every vector $t$ of constants in $L_1$, $M_1, w \models p(t)$ iff $M_1, w \models \tau(p(t))$, and therefore $M_1|_{L_1}, w \models p(t)$ iff $M_1|_{L_2}, w \models \tau(p(t))$. Analogously, $M_2|_{L_1}, w \models p(t)$ iff $M_2|_{L_2}, w \models \tau(p(t))$.
- $\Phi_\tau(M_1|_{L_2}) = \Phi_\tau(M_2|_{L_2})$: if $p(t)$ is an atomic sentence of $L_1$ and $w \in [h, t]$, then

$$\Phi_\tau(M_1|_{L_2}), w \models p(t) \iff M_1|_{L_2}, w \models \tau(p(t)). \quad \text{from (6)}$$

$$\iff M_1|_{L_1}, w \models p(t), \quad \text{previous item}$$

$$\iff M_2|_{L_1}, w \models p(t), \quad \text{by hypothesis}$$

$$\iff M_2|_{L_1}, w \models \tau(p(t)), \quad \text{previous item}$$

$$\iff \Phi_\tau(M_2|_{L_2}), w \models p(t), \quad \text{from (6).}$$

Thus $M_1|_{L_2} = M_2|_{L_2}$ for $\Phi_\tau$, so $\Phi_\tau$ is injective; therefore $M_1 = M_2$ and $\Pi$ defines implicitly the vocabulary of $L_2$.

Using the Beth property, we deduce that $\Pi$ defines explicitly the predicates, the constants and the functions of $L_2$, i.e. there exists an interpretation $\sigma$ of $L_2$ in $L_1$ such that:

$$\Pi \models \forall x(q(x) \leftrightarrow \delta_0^\Pi(x)),$$

$$\Pi \models \forall x(x \equiv a \leftrightarrow \delta_0^\Pi(x)),$$

$$\Pi \models \forall x \forall y(f(x) \equiv y \leftrightarrow \delta_0^\Pi(x, y)). \quad \text{(11)}$$

Now, we can establish claim (ii). Let $M$ be a model of $\Pi_2$ and let us consider the extension to the language $L_1 \cup L_2$ defined by $M^+$. $w \models p(t)$ iff $M, w \models \tau(p(t))$; so $M^+$ is a model of $\Pi$ and, by (11), it is also a model of every definition and thus, for every basic atom $q(t)$ in $L_2$ and $w \in [h, t]$,

$$M, w \models q(t) \iff M^+, w \models q(t)$$

$$\iff M^+, w \models \delta_0^\Pi(t), \quad \text{by (11)}$$

$$\iff M, w \models \tau(\delta_0^\Pi(t)), \quad \text{by definition of } M^+ \text{ and induction.}$$

By induction, for every $L_2$-formula $\psi(x)$, we deduce that $M, w \models \psi(t) \leftrightarrow \tau(\sigma(\psi(t)))$ and therefore $M \models \forall x(\psi(x) \leftrightarrow \tau(\sigma(\psi(x))))$. □

An interpretation satisfying any of (i)–(iii) of Proposition 7 is said to be surjective. Such interpretations preserve the property of being an equilibrium model, in the following sense.
Proposition 8. Let $\tau$ be a surjective interpretation of $\Pi_1$ in $\Pi_2$. For any model $\mathcal{M}$ of $\Pi_2$, if $\Phi_\tau(\mathcal{M})$ is an equilibrium model of $\Pi_1$ then $\mathcal{M}$ is an equilibrium model of $\Pi_2$.

Proof. Since $\tau$ is surjective, the map $\Phi_\tau$ is an injection of $\Pi_2$ models into $\Pi_1$ models. Applying the definition of equilibrium and Lemma 3 establishes the result. $\Box$

5. Synonymy and equivalence

We are now ready to consider our main concept. An interpretation $\tau$ that is both surjective and faithful is said to be a bijective interpretation of $\Pi_1$ in $\Pi_2$.

Definition 10 (Synonymy). If there exists a bijective interpretation $\tau$ of $\Pi_1$ in $\Pi_2$ (i.e., a surjective and faithful interpretation), we say that $\Pi_1$ is synonymous with $\Pi_2$ with respect to $\tau$.

It is easy to verify that if $\tau$ is a bijective interpretation of $\Pi_1$ in $\Pi_2$, then its inverse interpretation $\sigma$ is an interpretation of $\Pi_2$ in $\Pi_1$ and it is also bijective. In fact we have:

Theorem 2. Synonymy is an equivalence relation.

Proof.

- Reflexivity is trivial using $\tau = \text{id}$.
- To prove symmetry, let us consider a bijective interpretation $\tau$ of $\Pi_1$ in $\Pi_2$ and its inverse interpretation $\sigma$ of $L_2$ in $L_1$; to conclude symmetry it is enough to prove that $\sigma$ is an interpretation of $\Pi_2$ in $\Pi_1$ and that it is also bijective:
  - To prove that $\sigma$ is an interpretation of $L_2$ in $\Pi_1$ we must check the conditions (4) and (5) for $\sigma$. Let $a$ be a constant of $L_2$; applying the condition (ii) of Proposition 7 to the formula $\psi(x) = x \equiv a$ we have
    $$\Pi_2 \models \forall y (y \equiv a \leftrightarrow \tau(\delta_0^\tau(y)))$$
    and thus
    $$\Pi_2 \models \exists x \forall y (y \equiv x \leftrightarrow \tau(\delta_0^\tau(y))).$$
  - Then, by the faithfulness of $\tau$,
    $$\Pi_1 \models \exists x \forall y (y \equiv x \leftrightarrow \delta_0^{\tau^2}(y)),$$
    which is the condition (4) for $\sigma$. Condition (5) can be proved analogously.
- $\sigma$ is an interpretation of $\Pi_2$ in $\Pi_1$ and it is faithful:
  $$\Pi_1 \models \sigma(\varphi) \iff \Pi_2 \models \tau(\sigma(\varphi)) \quad (\text{because } \tau \text{ is faithful})$$
  $$\iff \Pi_2 \models \varphi \quad \text{(from Proposition 7 (ii) applied to } \tau)$$
- $\sigma$ is surjective: in particular, we can verify condition (ii) in Proposition 7 where $\tau$ is the inverse of $\sigma$:
  $$\Pi_2 \models \forall x (\tau(\sigma(x)) \equiv \tau(\sigma(\tau(x)))) \Rightarrow \Pi_2 \models \tau(\forall x (\sigma(x) \equiv \sigma(\tau(x))))$$
  $$\Rightarrow \Pi_1 \models \forall x (\sigma(x) \equiv \sigma(\tau(x))).$$
- Let us consider a bijective interpretation $\tau_1$ of $\Pi_1$ in $\Pi_2$ and a bijective interpretation $\tau_2$ of $\Pi_2$ in $\Pi_3$. To establish transitivity, we are going to prove that $\tau_2 \circ \tau_1$ is a bijective interpretation of $\Pi_1$ in $\Pi_3$.
  - $\tau_2 \circ \tau_1$ is an interpretation of $\Pi_1$ in $\Pi_3$ and it is faithful because $\tau_1$ and $\tau_2$ are:
    $$\Pi_1 \models \varphi \iff \Pi_2 \models \tau_1(\varphi) \iff \Pi_3 \models \tau_2(\tau_1(\varphi)).$$
  - $\tau_2 \circ \tau_1$ is surjective: in particular, $\Phi_{\tau_1} \circ \Phi_{\tau_2}$ verifies condition (iii) in Proposition 7, because $\Phi_{\tau_2}$ and $\Phi_{\tau_1}$ are injections. $\Box$

Proposition 9. If $\tau$ is a bijective interpretation of $\Pi_1$ in $\Pi_2$ then the mapping $\Phi_\tau$ is a one-one correspondence between models of $\Pi_1$ and models of $\Pi_2$. 

5.1. Verifying the adequacy conditions

Let us now consider synonymy in light of the adequacy conditions D1–D6. We have already dealt with the basic cases of translation and associated semantic correspondence. Let us now consider the sense in which two synonymous theories can be considered equivalent as in requirement D3.

Proposition 10. Let \( \Pi_1 \) and \( \Pi_2 \) be synonymous w.r.t. \( \tau \) and \( \sigma \). Then \( \Pi_2 \cup \tau \) is strongly equivalent with \( \Pi_1 \cup \sigma \). Thus \( \Pi_1 \) and \( \Pi_2 \) have a common definitional extension, i.e. there is a theory \( \Pi \) in \( L_2 \cup L_1 \), such that \( \Pi_2 \cup \tau \equiv \Pi_1 \cup \sigma \equiv \Pi \).

Proof. By Theorem 1, it is enough to prove that \( \Pi_2 \cup \tau \) and \( \Pi_1 \cup \sigma \) have the same models. Let \( M \) a model of \( \Pi_2 \cup \tau \). From \( M \models \forall x(p(x) \leftrightarrow \delta^\sigma_q(x)) \) it is easy to conclude by induction that

\[
    M \models \forall \exists \Longleftrightarrow M \models \tau(\varphi)
\]

for every \( L_1 \cup L_2 \)-sentence. By the definition of interpretation, \( M \models \tau(\varphi) \), for every \( \varphi \in \Pi_1 \) and thus \( M \models \varphi \) for every \( \varphi \in \Pi_1 \), i.e. \( M \models \Pi_1 \).

On the other hand, by item (ii) in Proposition 7 applied to \( \tau \), \( M \models \forall x(q(x) \leftrightarrow \tau(\delta^\sigma_q(x))) \) for all \( q \in P_2 \), and thus, for every vector \( t \) of constants in \( L_2 \), \( M \models q(t) \) iff \( M \models \tau(\delta^\sigma_q(t)) \), and iff \( M \models \delta^\sigma_q(t) \), by (12). Therefore \( M \models \forall x(q(x) \leftrightarrow \delta^\sigma_q(x)) \).

The proof of \( \Pi_1 \cup \sigma \equiv \Pi_2 \cup \tau \) is similar. \( \square \)

In fact Proposition 10 can be strengthened to an equivalence: two theories are bijectively interpretable if and only if they have a common definitional extension. This expresses one way in which the two theories are in an obvious sense equivalent once enriched with suitable translation manuals. Notice too that there is a close relationship between \( \Pi_2 \) and the translation \( \tau(\Pi_1) \) of \( \Pi_1 \) (similarly between \( \Pi_1 \) and the translation \( \sigma(\Pi_2) \) of \( \Pi_2 \)). It is already clear that \( \Pi_2 \models \tau(\Pi_1) \).

Although it is not generally true, even in the classical case, that \( \Pi_2 \models \tau(\Pi_1) \), we do however have:

Corollary 2. Let \( \Pi_1 \) and \( \Pi_2 \) be synonymous w.r.t. \( \tau \) and \( \sigma \). For any \( L_2 \)-formula \( \varphi \), \( \Pi_2 \models \varphi \iff \tau \sigma(\varphi) \), and \( \Pi_2 \models \varphi \Rightarrow \tau(\Pi_1) \models \tau \sigma(\varphi) \).

Next we turn to condition D4.

Proposition 11. Let \( \Pi_1 \) and \( \Pi_2 \) be theories in \( L_1 \) and \( L_2 \) respectively, synonymous w.r.t. \( \tau \) and \( \sigma \). Then the bijective mapping \( \Phi_\tau \) from models of \( \Pi_2 \) to models of \( \Pi_1 \) preserves the equilibrium property, i.e. \( \Pi_2 \models \tau \Pi_1 \).

Proof. Using Proposition 8, we only need to prove that if \( M \) is an equilibrium model of \( \Pi_2 \) then \( \Phi_\tau(M) \) is also in equilibrium. Let us assume that \( \Phi_\tau(M) \) is a model of \( \Pi_1 \) that is not in equilibrium. So there is \( M' \prec \Phi_\tau(M) \) with \( M' \models \Pi_1 \).

Since \( \Phi_\tau(\Phi_\sigma(M')) = M' \), by Lemma 3, \( \Phi_\sigma(M') \) is not total since \( M' \) is not total, and \( \Phi_\sigma(M') \prec \Phi_\sigma(\Phi_\tau(M)) = M \) which contradicts the equilibrium property of \( M \). \( \square \)

Clearly, condition D5 is satisfied and the presence of an inverse interpretation provides the sense in which the correspondence between \( \Pi_1 \) and \( \Pi_2 \) is idempotent. Lastly we consider D6.

Proposition 12. Let \( \Pi_1 \) and \( \Pi_2 \) be theories in \( L_1 \) and \( L_2 \) respectively synonymous w.r.t. \( \tau \) and \( \sigma \). Let \( \Pi \) a set of \( L_1 \)-formulas. Then \( \Pi_1 \cup \Pi \) is synonymous with \( \Pi_2 \cup \tau(\Pi) \) w.r.t. \( \tau \) and \( \sigma \).

Proof. Assume the hypotheses of the proposition. If \( \Pi_1 \cup \Pi \models \varphi \) and \( M \models \Pi_2 \cup \tau(\Pi) \), then \( \Phi_\tau(M) \models \Pi_1 \) and \( \Phi_\sigma(M) \models \Pi; \) thus \( \Phi_\tau(M) \models \varphi \) and \( M \models \tau(\varphi) \). Therefore \( \tau \) is an interpretation of \( \Pi_1 \cup \Pi \) in \( \Pi_2 \cup \tau(\Pi) \). Next we show that \( \tau \) is faithful, that is that

\[
    \Pi_2 \cup \tau(\Pi) \models \tau(\varphi) \Rightarrow \Pi_1 \cup \Pi \models \varphi.
\]

Now suppose that \( \Pi_2 \cup \tau(\Pi) \models \tau(\varphi) \) and \( M \models \Pi_1 \cup \Pi \). By assumption, \( M \models \Phi_\sigma(M) \). Since \( \sigma \) is an interpretation of \( \Pi_2 \) in \( \Pi_1 \), and \( M \models \Pi_2 \), \( \Phi_\sigma(M) \models \tau(\Pi) \) and \( \Phi_\sigma(M) \models \Pi_2 \). Therefore \( \Phi_\sigma(M) \models \tau(\varphi) \) and \( M \models \tau(\varphi) \). Finally \( \Phi_\sigma \) is injective over the set of models of \( \Pi_2 \) and therefore over the set of models of \( \Pi_2 \cup \tau(\Pi) \). So \( \tau \) is a bijective interpretation of \( \Pi_1 \cup \Pi \) onto \( \Pi_2 \cup \tau(\Pi) \). \( \square \)
6. A case study from causal action theory

To illustrate how our concepts can be applied in practice, we look at an example taken from causal action theory. This is a real application scenario where problems of equivalence arise in practice. The case study comes from [19] where Balduccini and Gelfond propose an architecture for a software agent that operates a physical device and is capable of monitoring, testing and repairing the device’s components. Based on the theory described in a suitable action language, they present simplified definitions of several notions, such as situation and diagnosis, in such a way that it is possible to give a simple account of the agent’s behaviour in which many of the agent’s tasks are reduced to computing the stable models of logic programs. We choose here the program built by the authors to find candidate diagnoses using answer set programming; to do that, a System Description of a Diagnostic Domain and a specific history is given. We only include here a short and informal description, for detailed explanations the reader should consult the referred paper [19].

We call \( L_2 \) the language used in [19] and we use \( \Pi_2 \) to denote our rendering of the program described in Section 4 of [19]. The language includes the natural numbers and the successor function; additionally, among the constants, we have a set of actions, a set of laws, a set of fluents and a function over the set of fluents: \( \ell \) is the negation of the fluent \( \ell \). Several predicates are used to fix the specific role of every constant and the basic relations among them:

\[
d_{\text{law}}(d) \quad (d \text{ is a dynamic causal law}),
\]

\[
s_{\text{law}}(d) \quad (d \text{ is a static causal law}),
\]

\[
\text{head}(d, l_0) \quad (\text{every law } d \text{ has assigned a list of fluents and } l_0 \text{ is the head}),
\]

\[
\text{action}(d, a) \quad (\text{the law } d \text{ specifies the effects of action } a).
\]

The fluents and the actions occur in specific instants and some predicates express that:

\[
\text{holds}(l, n) \quad (\text{the fluent } l \text{ holds at instant } n),
\]

\[
\text{obs}(l, n) \quad (\text{the fluent } l \text{ was observed to be true at moment } n),
\]

\[
\text{hpd}(a, n) \quad (\text{the action } a \text{ was observed to happen at moment } n),
\]

\[
\mathit{o}(a, n) \quad (\text{the action } a \text{ occurs at moment } n).
\]

Two relations, \( \text{hpd} \) and \( \mathit{o} \), are used to distinguish between actions observed and actions hypothesised respectively.

We re-write the rules of [19] as logical formulas separated by commas, understood to be universally closed (we omit the quantifiers for simplicity). Some formulas are included in order to describe the effects of causal laws and constraints, some establish the relationship between observations and the basic relations, and guarantee that observations do not contradict the agent’s expectations. The fifth formula rules out inconsistent states.

\[
\Pi_0 = \{ \neg(\text{holds}(l_1, x) \land \cdots \land \text{holds}(l_m, x) \land \mathit{o}(a, x)), \\
(d_{\text{law}}(x) \land \text{head}(x, y) \land \text{action}(x, z) \land \mathit{o}(z, u) \land \text{prec}_{\text{h}}(x, u)) \rightarrow \text{holds}(y, u + 1), \\
(s_{\text{law}}(x) \land \text{head}(x, y) \land \text{prec}_{\text{h}}(x, z)) \rightarrow \text{holds}(y, z), \\
(\text{holds}(y, z) \land \neg\text{holds}(\bar{y}, z + 1)) \rightarrow \text{holds}(y, z + 1), \\
\neg(\text{holds}(y, z) \land \text{holds}(\bar{y}, z)), \\
\text{hpd}(x, y) \rightarrow \mathit{o}(x, y), \\
\text{obs}(y, 0) \rightarrow \text{holds}(y, 0), \\
\neg(\mathit{o}(y, z) \land \neg\text{holds}(y, z)) \} \cup \text{SD}
\]

where the set \( \text{SD} \) includes the atoms \( d_{\text{law}}(d), \text{action}(d, a), s_{\text{law}}(d'), \text{head}(d, l_0), \text{action}(d, a) \) determined by the laws, actions and fluents defining the Description System, and \( \text{prec}_{\text{h}} \) is an auxiliary predicate which will be defined later.

The program \( \Pi_0 \) does not yet include the rules that we want to focus on in our example; in fact we consider the program \( \Pi_2 \) given by

\[
\Pi_2 = \Pi_0 \cup \{ \text{prec}_{\text{rei}}(d, 1, l_1), \ldots, \text{prec}_{\text{rei}}(d, m, l_m), \\
(\text{prec}_{\text{rei}}(x, u, v) \land \text{holds}(v, z) \land \text{all}_{\text{h}}(x, u + 1, z)) \rightarrow \text{all}_{\text{h}}(x, u, z), \\
\text{all}_{\text{h}}(x, np(x) + 1, z), \\
\text{all}_{\text{h}}(x, 1, z) \rightarrow \text{prec}_{\text{h}}(x, z) \}.
\]

The goal of these additional rules is to define the relation \( \text{prec}_{\text{h}} \): for every law \( d \), the atom \( \text{prec}_{\text{h}}(d, x) \) says that all the preconditions of \( d \) are satisfied at moment \( x \). This relation is defined via an auxiliary relation \( \text{all}_{\text{h}}(d, i, \ell) \) which holds if the
preconditions \(l_1, \ldots, l_m\) of \(d\) are satisfied at moment \(t\); additionally, we need the predicate \(\text{prec}_{\text{rei}}(d, n, l)\) to say that the \(n\)th precondition of \(d\) is \(l\) and the function \(\text{np}(d)\) that returns the number of preconditions of the law \(d\) and its interpretation is fixed.

Michael Gelfond has suggested the following problem.\(^{19}\) We know that it is also possible to manage the preconditions of a law using lists, though these are not available in most answer set solvers. Alternatively we can use the formulation without lists. The problem is to prove that both representation are equivalent.

We are going to introduce the alternative representation as a new theory \(\Pi_1\) in a language \(\mathcal{L}_1\) and then use the synonymy theory to answer the question. To approach this problem we consider a language \(\mathcal{L}_1\) that only differs from \(\mathcal{L}_2\) with respect to two predicates: \(\text{hold}(x, y)\)\(^{20}\) and \(\text{prec}(x, y)\) will be used to define \(\text{prec}_{\text{h}}\) instead of \(\text{all}_{\text{h}}(x, y, z)\), \(\text{prec}_{\text{num}}(x, y)\) and \(\text{prec}_{\text{rei}}(x, y, z)\). In the theory \(\Pi_1\) we will use the standard functions and predicates necessary to work with lists; we assume that these functions and predicates are available in \(\mathcal{L}_1\) and in \(\mathcal{L}_2\); moreover, we suppose that both theories include the necessary rules to define these predicates. So, \(\text{nil}\) denotes the empty lists and \(\text{cons}(x, y)\) is the list with head \(x\) and tail \(y\). The predicate \(\text{nth}(n, \ell, x)\) is read as “the \(n\)th term of \(\ell\) is \(x\)”; the predicate \(\text{member}(x, \ell)\) is read as “\(x\) is a member of \(\ell\)”; the predicate \(\text{length}(\ell, n)\) is read as “\(n\) is the length of \(\ell\)”; the predicate \(\text{append}(\ell_1, \ell_2, \ell)\) is read as “\(\ell_2\) is the concatenation of \(\ell_1\) and \(\ell_2\)”. In the following theory, \(\ell\) denotes the list \([l_1, \ldots, l_m]\)

\[
\Pi_1 = \Pi_0 \cup \{\text{hold}(\text{nil}, x),
\hspace{1cm}
  (\text{holds}(x, y) \land \text{hold}(z, y)) \rightarrow \text{hold}(\text{cons}(x, z), y),
\hspace{1cm}
  (\text{prec}(x, y) \land \text{hold}(y, z)) \rightarrow \text{prec}_{\text{h}}(x, z)\}.
\]

Here, we use the predicate \(\text{hold}(\{l_1 \ldots l_n\}, x)\) to say that every precondition in the list \([l_1 \ldots l_n]\) holds at instant \(x\) and \(\text{prec}(d, \{l_1 \ldots l_n\})\) to say that the preconditions of \(d\) are \([l_1 \ldots l_n]\).

Intuitively, the definitions of \(\text{prec}_{\text{h}}\) provided by the two programs \(\Pi_2\) and \(\Pi_1\) are essentially the same; but there is no standard technique in ASP to establish this. The theory of synonymy developed in this paper allows us to address this problem and provide a formal solution. Notice that we regard all the vocabulary that is common to the two programs as having the same meaning in each. So their translations are based on the identity map and are left implicit here. We only deal explicitly with the translation of those predicates that differ in \(\mathcal{L}_1\) and \(\mathcal{L}_2\). Specifically, we can define an interpretation \(\tau\) of \(\mathcal{L}_1\) to \(\mathcal{L}_2\) by:

\[
\begin{align*}
\text{prec}(x, y) & \iff \forall u \forall v (\text{nth}(u, y, v) \rightarrow \text{prec}_{\text{rei}}(x, u, v)), \\
\text{hold}(x, y) & \iff \forall u (\text{member}(u, x) \rightarrow \text{holds}(u, y)).
\end{align*}
\]

In addition, there is an interpretation \(\sigma\) of \(\mathcal{L}_2\) to \(\mathcal{L}_1\) defined by:

\[
\begin{align*}
\text{prec}_{\text{rei}}(x, y, z) & \iff \exists u (\text{nth}(y, u, z) \land \text{prec}(x, u)), \\
\text{all}_{\text{h}}(x, y, z) & \iff \exists u \exists v \exists w (\text{prec}(x, u) \land \text{length}(w, y-1) \land \text{append}(w, v, u) \land \text{hold}(v, z)).
\end{align*}
\]

To prove that \(\tau\) is an interpretation of \(\Pi_1\) in \(\Pi_2\), we must check that if \(I\) is a model of \(\Pi_2\) then \(I\) is a model of \(\tau(\varphi)\) for every \(\varphi \in \Pi_1\). We give the details for some formulas:

\begin{itemize}
  \item \(\tau(\text{prec}(d, \ell)) = \forall \forall y (\text{nth}(x, \ell, y) \rightarrow \text{prec}_{\text{rei}}(d, x, y))\). If \(I\) is a model of \(\Pi_2\), then, necessarily \(\text{nth}(i, \ell, l_i)\), with \(i = 1, \ldots, n\), are just the atoms with predicate \(\text{nth}\) and the list \(\ell\) in the central argument that are valid in \(I\), and \(I \models \text{prec}_{\text{rei}}(d, l_i, \ell)\) for every \(i = 1, \ldots, m\); therefore, \(I \models \forall \forall y (\text{nth}(x, \ell, y) \rightarrow \text{prec}_{\text{rei}}(d, x, y))\).
  \item \(\tau(\text{prec}_{\text{h}}(x, z)) =
\hspace{1cm}
  = (\forall v \forall u (\text{nth}(u, y, v) \rightarrow \text{prec}_{\text{rei}}(x, u, v)) \land \forall v (\text{member}(v, y) \rightarrow \text{holds}(v, z))) \rightarrow \text{prec}_{\text{h}}(x, z)
\hspace{1cm}
  \equiv \forall v \forall u ((\text{nth}(u, y, v) \rightarrow \text{prec}_{\text{rei}}(x, u, v)) \land (\text{member}(v, y) \rightarrow \text{holds}(v, z))) \rightarrow \text{prec}_{\text{h}}(x, z).
\]
\end{itemize}

This formula is trivially valid if \(y\) ranges from the set of lists of preconditions and thus we can assume that \(y = [v_1, \ldots, v_m]\) and the formula becomes equivalent to

\[
((\text{prec}_{\text{rei}}(x, 1, v_1) \land \text{holds}(v_1, z)) \land \cdots \land (\text{prec}_{\text{rei}}(x, m, v_m) \land \text{holds}(v_m, z))) \rightarrow \text{prec}_{\text{h}}(x, z),
\]

where \(m = \text{np}(x)\). This formula can be deduced from \(\Pi_2\); moreover, it could be easily deduced from the following subset:

\(^{19}\) In a personal communication to the authors.

\(^{20}\) Notice that the \text{hold} predicate is distinct from the \text{holds} predicate.
for all \( M \) in Proposition 7. This is easy to prove by considering that the translation between models. We do not yet have comparable results in the setting of ASP. However we may conjecture that interpretational equivalence of models, while definitions is less developed, however especially in the case of superintuitionistic logics much is known about key properties, such as recent account of translatability issues in such contexts. The theory of interpretations and equivalence in non-classical logics definitional equivalence was extended and applied to empirical forms of knowledge in [56,33,32]; see also [34] for a more does not give an adequate account of the translation of expressions with function symbols. Good textbook treatments and proof can be found in [36]. In our approach we have partly followed the style of Pinter in [36], however that paper 8. Literature and related work

An interesting area for future study is the way in which different kinds of interpretability or synonymy relations preserve properties, including properties of the theories and their models. The case in which interpretations in classical logic, many results are known (see e.g. [36]). Typical observations are that surjective interpretations \( \tau \) preserve the elementary equivalence of models, while definitions \( \sigma \) of a special syntactic kind can preserve properties like homomorphisms between models. We do not yet have comparable results in the setting of ASP. However we may conjecture that interpretations preserving the \( \sqsubseteq \)-relation between structures could be relevant in some cases for the transfer of certain computational properties from one theory to another.\(^{21}\)

8. Literature and related work

In classical logic there is a large and well-developed body of work on interpretability dating from the 1950s. The first systematic treatments of synonymous theories in this context can be found in [53,35], a more algebraic approach can be found in [54]. The classical version of Proposition 7 is essentially contained in [53], though a more detailed statement and proof can be found in [36]. In our approach we have partly followed the style of Pinter in [36], however that paper does not give an adequate account of the translation of expressions with function symbols. Good textbook treatments of interpretability can be found in [47,55]. Outside the field of mathematics, the classical theory of interpretability and definitional equivalence was extended and applied to empirical forms of knowledge in [56,33,32]; see also [34] for a more recent account of translatability issues in such contexts. The theory of interpretations and equivalence in non-classical logics is less developed, however especially in the case of superintuitionistic logics much is known about key properties, such as interpolation and Beth, on which interpretability theory depends, see e.g. [50,51,57]. In the context of nonmonotonic logic

\(^{21}\) This is related to the fact that for logic programs some syntactic classes can be characterised relative to others in terms of properties of the \( \sqsubseteq \)-relation [52].
programming the study of different kinds of equivalence between programs is relatively new (see references in Section 1).
Until now the case of programs in different languages has only been considered in [31]. There has been some discussion of
the role and properties of definitions in ASP in [58,59].

9. Concluding remarks

We have shown how formal approaches to intertheory relations developed for mathematical and scientific knowledge
can be applied to systems of logic programming and nonmonotonic reasoning used for practical problem solving and knowl-
gedge representation. In particular, we have described how the theory of interpretability and definitional equivalence can be
applied in the context of first-order logic programs under answer set semantics and nonmonotonic theories in the system
of quantified equilibrium logic. In this setting we regard theories as synonymous if each is bijectively interpretable in the
other, and we have characterised this relation in different ways. We also showed that this reconstruction satisfies a number
of intuitive, informal adequacy conditions.

Our approach should be directly applicable to all the usual systems of ASP as well as specialised languages such as
action languages built on them. In addition our general method should be transferable to other knowledge representation
formalisms based on logic. Specific characterisations of synonymy will depend on the logics concerned and their metalogical
properties.

The applicability of what is essentially a classical logical approach in a non-classical context relies on two essential
features: first, our underlying logic has several properties such as Beth that help to relate the syntax to the semantics of
definitions and translations; secondly, in ASP and equilibrium logic the strong concept of equivalence between theories is
fully captured in the underlying monotonic logic (quantified here-and-there). This allows us to define a robust or modular
concept of equivalence across different languages.

Several avenues are left open for future exploration. For example, one might want to study other kinds of interpretability
relations, such as where the formula \( \delta_\tau \) defining a predicate \( p \) may contain additional parameters, or where the semantic
mapping \( \varphi_\tau \) may relate models with different domains. Secondly, one might search for simple structural properties on the
models of two programs or theories that are equivalent to or sufficient for synonymy. Thirdly, based on these or other
properties of the theories concerned, it would be useful to develop systems for checking synonymy, thereby extending
current methods for checking strong equivalence in the case of programs in the same language [28,26].

Acknowledgments

We are very grateful to Michael Gelfond for suggesting the case study treated in Section 6 and for commenting on an
earlier draft of this paper. We would also like to thank Albert Visser for help in clarifying some issues in interpretability
theory and with the counterexample presented in Section 5. As always, Larisa Maksimova has been a friendly and vital
source of information about interpolation and Beth properties in non-classical logics. Last, we would like to express our
thanks to two anonymous referees for their careful reading of the paper and many suggestions for improvements, as well
as to Luigi Palopoli for his invitation to contribute to this special issue. Any remaining errors or omissions are due either
to the first or the second author.

References

2003, pp. 63–74.
2006.