From description-logic programs to multi-context systems

Luís Cruz-Filipe[a,*], Graça Gaspar[b], Isabel Nunes[b]

[a] *Department of Mathematics and Computer Science, University of Southern Denmark*
[b] *BioISI—Biosystems & Integrative Sciences Institute, Faculty of Sciences, University of Lisbon*

Abstract

The combination of logic program-style rules with other reasoning systems has been a fertile topic of research in the last years, with the proposal of several different systems that achieve this goal. In this work, we look at two of these systems, dl-programs and multi-context systems, which address different aspects of this combination, and include different, incomparable programming constructs. We prove that every dl-program can be transformed into a multi-context system in such a way that the different semantics for each paradigm are naturally related. As a consequence, constructions developed for dl-programs can be automatically ported to multi-context systems. In particular, we show how to model default rules over ontologies with the usual semantics.

Keywords: description logics; logic programming; semantics; rule-based systems; formal translations

1. Introduction

Several approaches combining rules and ontologies have been proposed in the last years for semantic web reasoning, e.g. [4, 14, 17, 18, 28, 35]. Ontologies are typically expressed through decidable fragments of function-free first-order logic with equality, offering a very good ratio expressiveness/complexity of reasoning [2]. By extending ontologies with rule capabilities, together with non-monotonic features (in particular, the negation-as-failure operator from logic programming), one obtains a very powerful framework for semantic web reasoning where it is possible to write more expressive queries.

In this paper, we look at two of these systems: dl-programs [17, 18] and multi-context systems (MCSs) [4], which address different aspects of this combination, and include incomparable programming constructs. These two paradigms differ in their essence – a dl-program is essentially a logic program that can query a single description logic knowledge base $\mathcal{L}$ and may "feed" its view of $\mathcal{L}$ with newly inferred facts, while MCSs consist of several knowledge bases, possibly expressed in different languages, each declaring additional rules that allow communication with the others.

We show that every dl-program can be transformed into a multi-context system. Although this transformation has been informally described earlier [5], the contribution of this work is not only making it precise, but also studying its theoretical properties, namely regarding semantic aspects, and discussing some practical implications. To the authors' knowledge, these aspects have never been addressed before. In particular, we show that our translation preserves the various existing semantics for each paradigm: answer sets for dl-programs become grounded equilibria for multi-context systems, whereas the well-founded semantics of a dl-program corresponds to the notion of well-founded belief set of a multi-context system. These results rely on a new definition of how to view a description logic knowledge base as a context of a multi-context system.

Translations between different knowledge representation formalisms have been studied by several authors [20, 29, 32, 36]. Such translations formally establish a precise relationship between the expressive

*Corresponding author; tel. +45 6550 2387, fax +45 6550 2373
Email addresses: lcf@imada.sdu.dk (Luís Cruz-Filipe), gg@di.fc.ul.pt (Graça Gaspar), in@di.fc.ul.pt (Isabel Nunes)

power of two formalisms. More importantly, they allow for the reuse of tools and techniques developed for the more expressive formalism to the less expressive one, by means of the translation. In our case, our translation makes it possible to apply reasoners for multi-context systems also to the translation of dl-programs, without needing dedicaded tools to deal with dl-programs. This is also the idea behind the implementation of the DL-plugin for `dlvhex` [21], which uses a translation of dl-programs into HEX-programs. On the other direction, constructions that were originally proposed for the less expressive formalism can be analyzed into the more general one by means of the translation. We illustrate this potential by showing how the original encoding of default rules in dl-programs [18] can be directly translated into MCSs, yielding a correspondence between extensions in default logic and equilibria in MCSs. Likewise, this translation could be applied e.g. to the systematic constructions for dl-programs described in [10] to yield similar techniques for MCSs.

The structure of the paper is as follows. Section 2 summarizes previous work on dl-programs and introduces our running example. Section 3 introduces multi-context systems and presents a translation from dl-programs to these, together with proofs of equivalence between the different semantics of both systems. Section 4 discusses in more detail two aspects of the translation: how to extend it to a more expressive variant of dl-programs; and how to use it to extend the encoding of default rules in dl-programs from [18] to the framework of MCSs. Section 5 concludes the presentation.

This article extends work previously published in [9, 11], namely by including the proofs of all results and a more detailed formal analysis of the design options and their consequences.

1.1. Related work

Combining description logics (DLs) with rules has been a leading topic of research for the past 15 years.

Description logic programs, or dl-programs [17, 19], are heterogeneous combinations of rules and ontologies, which are loosely coupled: the set of rules and the ontology are kept separate. The connection between these two components is achieved through the inclusion of *dl-atoms* in rule bodies, which perform queries to the knowledge base, possibly modifying it (adding facts) for the duration/purpose of each individual query. In particular, [17, 19] illustrate this approach for the logics of OWL-LITE and OWL-DL. The semantics of dl-programs extends two different semantics for ordinary normal logic programs – answer-set semantics and well-founded semantics.

A number of encouraging results for dl-programs are also presented in [17]: (i) data complexity is preserved for dl-programs reasoning under well-founded semantics as long as dl-queries in the program can be evaluated in polynomial time, and (ii) reasoning for dl-programs is first-order rewritable (and thus in LOGSPACE under data complexity) when the evaluation of dl-queries is first-order rewritable and the dl-program is acyclic. These nice results are a consequence of the use of Datalog: complexity of query evaluation on both Datalog and stratified Datalog$^\neg$ programs is data complete for PTIME and program complete for EXPTIME, as shown in [12].

There are other formalisms that allow combination of different reasoning systems. Hybrid MKNF knowledge bases [32] are a homogeneous approach, in the sense that the rules and the description logic knowledge base are not kept separate, as in dl-programs; for this reason, this formalism lacks modularity. On the other hand, various heterogeneous proposals have emerged to cope with several components, allowing to deal with information coming from different sources. HEX-programs [20], which generalize dl-programs, are higher-order logic programs with external atoms that may pose queries to systems using different languages.

In contrast with these approaches, where communication between system components is centralized in the logic program part of the system, multi-context systems [4] (MCSs) were originally designed to bring together characteristics of both heterogeneous monotonic [24, 31] and homogeneous non-monotonic systems [7, 34], capitalizing on both worlds.

Multi-context systems are similar to HEX-programs, in that they are heterogeneous non-monotonic systems, whose components (called *contexts*) are knowledge bases that can be expressed in different logics (e.g., a theory in classical logic, a description logic knowledge base, a set of modal or temporal logic formulas, or a non-monotonic formalism such as a logic program under answer set semantics, or a set of default logic rules). Unlike HEX-programs, however, MCSs' inter-component communication is distributed among the contexts via sets of (non-monotonic) *bridge rules*.

Since they were originally proposed, several variations of MCSs have been studied that add to their potential fields of application. Managed MCS [5] generalize bridge rules by allowing arbitrary operations on contexts (e.g. deletion or revision operators) to be freely defined. Relational MCSs [22] introduce variables and aggregate expressions in bridge rules, allowing a formal first-order syntax, and extending the semantics of MCSs accordingly. Dynamic MCSs [13] are designed to cope with situations where knowledge sources and their contents may change over time and are not known *a priori*. Evolving MCSs [26] extend this idea by incorporating knowledge resulting from dynamic observations through different belief change operations with different levels of persistence. This framework is further extended in [25] with the notion of evolving bridge rules.

A different line of research addresses techniques for dealing with inconsistency (non-existence of a model) within MCSs. The authors of [15] propose a declarative policy language, together with methodologies to apply it, to provide a means to create policies to avoid or repair inconsistencies in MCSs in a controlled way (e.g. specifying which inconsistencies can be repaired automatically, and which ones need external input by a human operator).

(Partial) translations between formalisms are an important tool to compare their expressive power and to allow transfer of technology from one formalism into another. Thus, hybrid MKNF knowledge bases can be translated into MCSs [29], providing a way for agents to reason with the former without the need for specialized Hybrid MKNF reasoners, whereas MCSs and HEX-programs are essentially incomparable [20]. In turn, the semantics of dl-programs has been captured within the framework of several, well-known, nonmonotonic logics, e.g. Autoepistemic Logic [8], Equilibrium Logic [23] (a logic-based version of Answer Set Semantics), Reiter's Default Logic [36], and MKNF [32]. Our results enrich this line of work by formally showing that dl-programs can be embedded into MCSs via a semantics-preserving translation.

Using our translation, we show how we can adapt the encoding of default rules in dl-programs from [18] to the more general scenario of multi-context systems, and illustrate our construction for the particular case of ontologies viewed as contexts. Our syntax of default rules is inspired by that of [3], although with some restrictions dictated by the limitations as to what we can write in the syntax of MCSs. We take Reiter's characterization of extensions as fixpoints as the definition in the ontology scenario, again following [3], and show that extensions can be characterized as particular models of the corresponding MCS.

Several frameworks for combining reasoning systems have been implemented. In particular, `dlvhex`, available at `http://www.kr.tuwien.ac.at/research/systems/dlvhex/`, is a prototype application for computing models of HEX-programs [20]. Its DL-Plugin simulates the behaviour of dl-programs by means of a rewriter that processes the syntax of dl-atoms, allowing the use of `dlvhex` directly as a reasoner for dl-programs. Both `dlvhex` and the DL-Plugin are implemented in C++, using RACER [27] as a DL reasoning engine to process OWL-DL ontologies.

A prototypical implementation of a fully distributed algorithm to compute semantics of MCSs, `dmcs`, is also available at `http://sourceforge.net/projects/dmcs/`. Another plug-in for `dlvhex`, MCS-IE (MCS Inconsistency Explainer), is also available at `http://www.kr.tuwien.ac.at/research/systems/mcsie/`. This tool provides reasons for inconsistency in MCSs as described in [4].

Several other mechanisms for combining rules and ontologies have been proposed, but are outside the scope of this paper. The interested reader can find a more comprehensive overview of the different approaches in the introduction of [32].

2. Description logic programs

Description logic programs (dl-programs) are at the core of the work presented in this article. This section summarizes the most relevant concepts and results about them, originally published in [17, 18], together with an example that is used throughout.

2.1. Description logics and ontologies

Description logics (DLs) are extensively used to express ontologies in the Semantic Web due to their nice behaviour – almost all of them are decidable fragments of function-free first-order logic with equality, offering a very good ratio expressiveness/complexity of reasoning [2].

In description logics, predicates are restricted to having arity at most 2; unary predicates are called *concepts* and binary predicates are called *roles*. Concepts and roles can be atomic or built using concept and role *constructors*. Each description logic allows a particular set of constructors, which determines its expressivity and complexity of reasoning.

We detail the concept constructors we use in our examples. Given concepts C and D and a role R, $\overline{C}$ is the *complement* of C; $C \sqcap D$ and $C \sqcup D$ are the intersection and union of C and D, respectively; $\{a_1, \ldots, a_n\}$ is the set containing exactly $a_1, \ldots, a_n$; $\exists R.C$ denotes the concept of being in relation R with another element satisfying concept C (in mathematical notation: $\{x \mid \exists y.R(x,y) \wedge C(y)\}$); $\geq n.R$ and denotes the concept of being in relation R with at least n other elements; $\bot$ is the empty concept.

The only role construct we use is inverse: R^R denotes the inverse relation of R. Other typical role constructs include intersection and union (defined as for roles), composition and transitive closure.

An *ontology* is a set of axioms in a description logic. These axioms are typically divided in two groups: *terminological* axioms (which are collectively called the *T-Box*) and *assertion* axioms (the *A-Box*). Terminological axioms express inclusions between concepts C and D and roles R and S, written $C \sqsubseteq D$ or $R \sqsubseteq S$; equivalence ($C \equiv D$ or $R \equiv S$) is shorthand for inclusion in both directions. Assertion axioms are of the form $C(a)$ or $R(a,b)$, where a and b are constants. Since description logics do not follow the Unique Name Assumption (distinct constants correspond to distinct individuals), we can also have assertions $a = b$ and $a \neq b$ between individuals.

In this work, we also refer to ontologies as *description logic knowledge bases* (or simply "knowledge bases"), and ignore the distinction between the T-Box and the A-Box.

Example 1. *We introduce our running example: a travel ontology Σ containing a series of travel-related concepts. This ontology is a restriction of the Travel Ontology* `travel.owl`*, available at* `http://protege.stanford.edu/ontologies/travel.owl`*.*

*The concepts in Σ are mostly related to destinations (*Destination*) and particular types of destinations (*Beach*, *UrbanArea*, *RuralArea*, *City*, *Capital*, *NationalPark*, *Town*, *Farmland*, *QuietDest*, *FamilyDest*, *BackpackersDest*, *BudgetHotelDest*). There are also some concepts related to activities (*Hiking*, *Museum*, *Sports*, *Adventure*) and types of accommodation that may be offered at different destinations (*Campground*, *Hotel*, *LuxuryHotel*, *BudgetAccomm*). The roles in Σ are* hasActivity*,* hasAccommodation*,* hasPart*.*

The relation among the different types of destinations are expressed in the T-Box by means of the following axioms.

$$\begin{array}{rrr}
\mathsf{RuralArea} \sqsubseteq \mathsf{Destination} & \mathsf{Farmland} \sqsubseteq \mathsf{RuralArea} & \mathsf{NationalPark} \sqsubseteq \exists \mathsf{hasActivity.Hiking} \\
\mathsf{UrbanArea} \sqsubseteq \mathsf{Destination} & \mathsf{NationalPark} \sqsubseteq \mathsf{RuralArea} & \mathsf{NationalPark} \sqsubseteq \exists \mathsf{hasAccommodation.Campground} \\
\mathsf{Beach} \sqsubseteq \mathsf{Destination} & \mathsf{City} \sqsubseteq \mathsf{UrbanArea} & \mathsf{City} \sqsubseteq \exists \mathsf{hasAccommodation.LuxuryHotel} \\
\mathsf{RuralArea} \sqcap \mathsf{UrbanArea} \sqsubseteq \bot & \mathsf{Town} \sqsubseteq \mathsf{UrbanArea} & \mathsf{Capital} \sqsubseteq \exists \mathsf{hasActivity.Museum} \\
 & \mathsf{Capital} \sqsubseteq \mathsf{City} &
\end{array}$$

According to the interpretation of concept constructs, axiom $\mathsf{RuralArea} \sqcap \mathsf{UrbanArea} \sqsubseteq \bot$ *expresses that* RuralArea *and* UrbanArea *are disjoint (their intersection is included in the emptyset), while axiom* $\mathsf{City} \sqsubseteq \exists \mathsf{hasAccommodation.LuxuryHotel}$ *states that every* City *is related by* hasAccommodation *to a* LuxuryHotel*.*

Some concepts can be defined by more complex formulas.

$$\begin{aligned}
\mathsf{QuietDest} &\equiv \mathsf{Destination} \sqcap \overline{\mathsf{FamilyDest}} \\
\mathsf{BackpackersDest} &\equiv \mathsf{Destination} \sqcap (\exists \mathsf{hasAccommodation.BudgetAccomm}) \sqcap (\exists \mathsf{hasActivity.}(\mathsf{Sports} \sqcup \mathsf{Adventure})) \\
\mathsf{FamilyDest} &\equiv \mathsf{Destination} \sqcap (\geq 1.\mathsf{hasAccommodation}) \sqcap (\geq 2.\mathsf{hasActivity}) \\
\mathsf{BudgetHotelDest} &\equiv \mathsf{Destination} \sqcap (\exists \mathsf{hasAccommodation.}(\mathsf{BudgetAccomm} \sqcap \mathsf{Hotel}))
\end{aligned}$$

The A-Box contains the following instances.

RuralArea(Woomera)	Town(Coonabarabran)	NationalPark(BlueMountains)
Capital(Canberra)	NationalPark(Warrenbungles)	RuralArea(CapeYork)
Capital(Sydney)	hasAccommodation(Sydney, FourSeasons)	LuxuryHotel(FourSeasons)
hasPart(Sydney, BondiBeach)	hasPart(Sydney, CorrawongBeach)	Beach(BondiBeach)
Beach(CorrawongBeach)	City(Cairns)	

We assume that Σ does not contain any individuals not included in this list.

2.2. Syntax of dl-programs

Dl-programs extend logic programs with special atoms that communicate with an external description logic knowledge base (hereafter simply called "knowledge base"), in the spirit of other generalizations of logic programs. The key ingredient of dl-programs is the notion of *dl-atom*. A dl-atom over a knowledge base Σ and a function-free first-order signature Φ has the form

$$DL[P_1\ op_1\ p_1, \ldots, P_m\ op_m\ p_m; Q](\bar{t})$$

where $m \geq 0$, each P_i is either a concept or role of Σ, or a special symbol in $\{=, \neq\}$; $op_i \in \{\uplus, \cup, \cap\}$; each p_i is a unary or binary predicate symbol of Φ, according to whether the corresponding P_i is a concept or a role; and $Q(\bar{t})$ is a *dl-query*, that is, it is either a concept inclusion axiom F or its negation $\neg F$, or of the form $C(t)$, $\neg C(t)$, $R(t_1, t_2)$, $\neg R(t_1, t_2)$, $=(t_1, t_2)$, $\neq(t_1, t_2)$, where C is a concept, R is a role, t, t_1 and t_2 are terms – constants from either Σ or Φ, or variables. The sequence $P_1\ op_1\ p_1, \ldots, P_m\ op_m\ p_m$ is the *input context* of the dl-atom; we use the greek letter χ to denote generic input contexts.

A *dl-rule* over Σ and Φ is a clause that may contain dl-atoms, i.e. it has the form

$$a \leftarrow b_1, \ldots, b_k, \mathbf{not}\ \ b_{k+1}, \ldots, \mathbf{not}\ \ b_m$$

where a is a logical atom and $b_1, \ldots, b_m$ are either logical atoms or dl-atoms – where the logical atoms are again built using terms from Φ and constants from Σ. As usual, the *head* of r is a and the *body* of r is $b_1, \ldots, b_k, \mathbf{not}\ \ b_{k+1}, \ldots, \mathbf{not}\ \ b_m$. A *dl-program* over Φ is a pair $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ where Σ is a knowledge base and $\mathcal{P}$ is a finite set of dl-rules over Σ and Φ (also referred to, in this context, as a *generalized logic program*); the signature Φ is usually left implicit. Negation in $\mathcal{P}$ is the usual negation-as-failure, whereas Σ has the usual open-world semantics of description logic knowledge bases.

The operators $\uplus$, $\cup$ and $\cap$ are used to extend Σ in the context of the current dl-query. Intuitively, $P_k \uplus p_k$ (resp. $P_k \cup p_k$) adds to P_k (resp. $\neg P_k$) all instances of (pairs of) terms for which p_k holds – the *extension* of p_k –, before evaluating the query. Likewise, $P_k \cap p_k$ adds to $\neg P_k$ all instances of terms for which p_k does not hold. This intuitive meaning of these operators is made precise in the formal semantics given below. These extensions do not change Σ, only affecting $\mathcal{P}$'s current view of Σ. Therefore, the two components are kept separate, giving dl-programs nice modularity properties, and there is a bidirectional flow of information via dl-atoms.

The operator $\cap$ makes the theory of dl-programs more complex, and for this reason it was not always included in subsequent work on this topic. For the sake of simplicity, we also omit it in the remainder of this section and in the presentation of the translation to multi-context systems in the next section. In Section 4.1, we show how our results can be generalized to the full language of dl-programs including $\cap$.

We adopt some notational conventions throughout this paper. Variables are capital letters in math font (X), while constants and terms are in sans serif. Predicate names (from the generalized logic program) always begin with a lowercase letter, while concepts and roles (from the knowledge base) are written exactly as they are defined in the source ontologies (we use an example ontology that is freely available online). We do *not* use different fonts for objects of $\mathcal{P}$ and objects of Σ, since these sets are not necessarily disjoint (the constants of Σ may be used in $\mathcal{P}$); for the sake of clarity, in examples we however abstain from using the same name for a predicate in $\mathcal{P}$ and a concept or role in Σ.

Example 2. *We introduce our running example of dl-program. The ontology Σ is the travel ontology from Example 1. The generalized logic program $\mathcal{P}$ extends Σ with three predicates, defined by the following rules.*

$$\mathsf{wineDest}(\mathsf{Tasmania}) \leftarrow \qquad (r_1)$$
$$\mathsf{wineDest}(\mathsf{TamarValley}) \leftarrow \qquad (r_2)$$
$$\mathsf{wineDest}(\mathsf{Sydney}) \leftarrow \qquad (r_3)$$

$$\mathsf{overnight}(X) \leftarrow DL[;\mathsf{hasAccommodation}](X,Y) \qquad (r_4)$$
$$\mathsf{oneDayTrip}(X) \leftarrow DL[\mathsf{Destination} \uplus \mathsf{wineDest}; \mathsf{Destination}](X), \mathbf{not}\ \mathsf{overnight}(X) \qquad (r_5)$$

The interesting rule is rule (r_5), which identifies destinations that are only suitable for one-day trips. These destinations are selected not only from the information originally in Σ, but by (i) extending the concept Destination *of Σ with all instances of* wineDest *from $\mathcal{P}$ and then (ii) querying this extended view of Σ for all instances of* Destination*. The three instances of* wineDest *correspond to three wine regions that are interesting tourist destinations. The result is then filtered using the predicate* overnight *defined in (r_4) as the set of destinations for which some accommodation is known. This rule uses role* hasAccommodation *of Σ, where* hasAccommodation*(t_1,t_2) holds whenever t_1 is a* Destination *and t_2 an accommodation facility located in t_1. The reason for resorting to (r_4) is the usual one: the operational semantics of negation-as-failure requires that all variables in a negated atom also appear non-negated in the body of the same rule.*

Since Sydney *is an individual of Σ, the first three rules have different impacts on the program: while (r_1) and (r_2) add new constants to the domain of $\mathcal{KB}$, rule (r_3) adds* information *about an individual already in Σ. In this particular case,* Sydney *is also an instance of* Destination *in Σ, so the input context in (r_5) actually adds only two new instances to this concept. Without the extended query in (r_5), we would not be able to infer e.g.* oneDayTrip*(*Tasmania*) without modifying ontology Σ.*

2.3. Semantics of dl-programs

Semantics for dl-programs use the notion of Herbrand base of a logic program $\mathcal{P}$ over Φ, denoted $\mathsf{HB}_\mathcal{P}$ – the set of all ground atoms consisting of predicate symbols and terms from Φ. Since Φ does not contain function symbols, $\mathsf{HB}_\mathcal{P}$ is always finite. The Herbrand base of a dl-program $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$, denoted $\mathsf{HB}_{\mathcal{KB}}$, is similarly defined, except that constant symbols may also come from the vocabulary of Σ, and is likewise always finite.

A subset I of $\mathsf{HB}_{\mathcal{KB}}$ is called an *interpretation* of $\mathcal{KB}$ relative to $\mathcal{P}$. We say that I is a *model* of a ground atom or dl-atom a under Σ, or I *satisfies* a under Σ, denoted $I \models_\Sigma a$, in the following cases:

- if $a \in \mathsf{HB}_{\mathcal{KB}}$, then $I \models_\Sigma a$ iff $a \in I$;
- if a is a ground dl-atom $DL[\chi; Q](\overline{t})$ with $\chi = P_1\ op_1\ p_1, \ldots, P_m\ op_m\ p_m$, then $I \models_\Sigma a$ iff $\Sigma(I;\chi) \models Q(\overline{t})$, where $\models$ denotes usual first-order entailment, $\Sigma(I;\chi) = \Sigma \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leq i \leq m$,

$$A_i(I) = \begin{cases} \{P_i(\overline{e}) \mid p_i(\overline{e}) \in I\}, & op_i = \uplus \\ \{\neg P_i(\overline{e}) \mid p_i(\overline{e}) \in I\}, & op_i = \uplus \end{cases}$$

An interpretation I is a *model* of a ground dl-rule r iff $I \models_\Sigma H(r)$ whenever $I \models_\Sigma l$ for all $l \in B(r)$, where $H(r)$ and $B(r)$ are the head and the body of rule r, respectively; I is a model of $\mathcal{KB}$ iff I is a model of all ground instances of the rules of $\mathcal{P}$. See Section 4.1 for the extension of these semantics to the full language including $\curlywedge$.

Example 3. *The Herbrand base of the dl-program $\mathcal{KB}$ of Example 2 contains all ground atoms built from applying* wineDest*,* overnight *and* oneDayTrip *not only to the constants of $\mathcal{P}$ –* Tasmania*,* TamarValley *and*

Sydney – *but also to all individuals of* Σ. *Thus,* $\mathsf{HB}_{\mathcal{KB}}$ *contains e.g.*

$\mathsf{wineDest}(\mathsf{Tasmania})$	$\mathsf{overnight}(\mathsf{Tasmania})$	$\mathsf{oneDayTrip}(\mathsf{Canberra})$
$\mathsf{wineDest}(\mathsf{FourSeasons})$	$\mathsf{overnight}(\mathsf{Sydney})$	$\mathsf{oneDayTrip}(\mathsf{Sydney}), \ldots$

The presence of e.g. $\mathsf{wineDest}(\mathsf{FourSeasons})$ *does not fit well with the intended interpretation of* $\mathsf{wineDest}$*; but this is a well-known side-effect of the absence of types in logic programming.*

Examples of interpretations for $\mathcal{KB}$ *are:*

$$
\begin{aligned}
I_1 &= \{\mathsf{wineDest}(\mathsf{Tasmania}), \mathsf{wineDest}(\mathsf{Sydney}), \mathsf{wineDest}(\mathsf{TamarValley})\} \\
I_2 &= \{\mathsf{wineDest}(\mathsf{Tasmania}), \mathsf{wineDest}(\mathsf{Sydney}), \mathsf{wineDest}(\mathsf{TamarValley})\} \cup \\
&\quad \cup \{\mathsf{oneDayTrip}(\mathsf{Tasmania}), \mathsf{oneDayTrip}(\mathsf{TamarValley})\} \cup \\
&\quad \cup \{\mathsf{oneDayTrip}(t) \mid \Sigma \models \mathsf{Destination}(t), t \neq \mathsf{Sydney}\} \cup \{\mathsf{overnight}(\mathsf{Sydney})\}
\end{aligned}
$$

To understand whether these interpretations are models of $\mathcal{KB}$*, we recall that the only instance of* $\mathsf{hasAccommodation}$ *in* Σ*, is* $\mathsf{hasAccommodation}(\mathsf{Sydney}, \mathsf{FourSeasons})$*. Therefore,* I_1 *is not a model of* $\mathcal{KB}$*, since it does not satisfy rule* (r_4) *with* $X = \mathsf{Sydney}$*, but* I_2 *is a model of* $\mathcal{KB}$*.*

A dl-program $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ is *positive* if the rules in $\mathcal{P}$ do not contain negations. Positive dl-programs enjoy the usual properties of positive logic programs, namely they have a unique least model $\mathsf{M}_{\mathcal{KB}}$ that can be constructed by computing the least fixed-point of the Herbrand transformation $\mathsf{T}_{\mathcal{KB}}$, defined as the usual one-step consequence operator for logic programs, resorting to Σ to evaluate dl-atoms. The dl-program in Example 2 is not a positive program because of the negation in rule (r_5).

Answer-set semantics. The answer-set semantics of dl-programs is defined again in analogy to that of logic programs. The presence of $\curlywedge$ allows for two possible generalizations [18]; since we omit this operator at this stage, we only present what those authors called "strong answer-set semantics", and omit the adjective "strong".

Given a dl-program $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$, we can obtain a positive dl-program by replacing $\mathcal{P}$ with its *dl-transform* $s\mathcal{P}^I_\Sigma$ relative to Σ and an interpretation $I \subset \mathsf{HB}_{\mathcal{KB}}$. This is obtained by grounding every rule in $\mathcal{P}$ and then (i) deleting every dl-rule r such that $I \models_\Sigma a$ for some default negated a in the body of r, and (ii) deleting from each remaining dl-rule the negative body. The informed reader will recognize this to be a generalization of the Gelfond–Lifschitz reduct. Since $\mathcal{KB}^I = \langle \Sigma, s\mathcal{P}^I_\Sigma \rangle$ is a positive dl-program, it has a unique least model $\mathsf{M}_{\mathcal{KB}^I}$. An *answer set* of $\mathcal{KB}$ is an interpretation I that coincides with $\mathsf{M}_{\mathcal{KB}^I}$.

Example 4. *Consider again the dl-program* $\mathcal{KB}$ *and the interpretations defined above. We can verify that* I_2 *is an answer set for* $\mathcal{KB}$*.*

First, we need to compute $s\mathcal{P}^{I_2}_\Sigma$*. Rules* (r_1)*,* (r_2) *and* (r_3) *are already ground rules, so they are included in* $s\mathcal{P}^{I_2}_\Sigma$ *unchanged. For rule* (r_4)*,* $s\mathcal{P}^{I_2}_\Sigma$ *contains all of its ground instances (obtained by replacing* X *and* Y *by all constants from both* Σ *and* $\mathcal{P}$*).*

Consider now the ground instances of (r_5)*. The only one case when* $I_2 \models \mathsf{overnight}(X)$ *requires* X *to be* Sydney*, so this instance is removed from* $s\mathcal{P}^{I_2}_\Sigma$*, while* $\mathbf{not}\, \mathsf{overnight}(t)$ *is removed from all other closed instances of* (r_5)*. Thus,* $s\mathcal{P}^{I_2}_\Sigma$ *contains* (r_1)*,* (r_2) *and* (r_3)*, together with*

$$
\begin{aligned}
&\mathsf{overnight}\,(t_1) \leftarrow DL[;\mathsf{hasAccommodation}]\,(t_1, t_2) && \text{for every } t_1, t_2 \in \mathsf{HB}_{\mathcal{KB}} \\
&\mathsf{oneDayTrip}(t) \leftarrow DL[\mathsf{Destination} \uplus \mathsf{wineDest}; \mathsf{Destination}](t) && \text{for } t \in \mathsf{HB}_{\mathcal{KB}} \setminus \{\mathsf{Sydney}\}
\end{aligned}
$$

Now we can compute the least model of $s\mathcal{P}^{I_2}_\Sigma$ *by iterating the Herbrand transform, and check that the result coincides with* I_2*. Therefore,* I_2 *is an answer set for* $\mathcal{KB}$ – *actually the only one, as we show below.*

Well-founded semantics. The well-founded semantics for logic programs can also be generalized to dl-programs [17], but only when $\not\in$ is absent. There are several equivalent ways to define this semantics; in this paper, we define it again by means of the Herbrand transform. For a dl-program $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ define $\gamma_{\mathcal{KB}}(I)$ to be the least model of the positive dl-program $\mathcal{KB}^I$ defined above. The operator $\gamma_{\mathcal{KB}}$ is anti-monotonic (if $I \subseteq J$, then $\gamma_{\mathcal{KB}}(I) \supseteq \gamma_{\mathcal{KB}}(J)$), so $\gamma^2_{\mathcal{KB}}$ is monotonic and therefore it has least and greatest fixpoints, denoted $\mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right)$ and $\mathsf{gfp}\left(\gamma^2_{\mathcal{KB}}\right)$, respectively. An atom $a \in \mathsf{HB}_{\mathcal{P}}$ is *well-founded* if $a \in \mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right)$ and *unfounded* if $a \notin \mathsf{gfp}\left(\gamma^2_{\mathcal{KB}}\right)$; the *well-founded semantics* of $\mathcal{KB}$ is the set containing all well-founded atoms (which are true in every model of $\mathcal{KB}$) and the negations of all unfounded atoms (which are false in every model of $\mathcal{KB}$). Unlike answer sets, the well-founded semantics of $\mathcal{KB}$ needs not be a model of $\mathcal{KB}$.

Example 5. *We quickly illustrate the essential steps in computing the well-founded semantics of the dl-program $\mathcal{KB}$ from Example 2.*

First, we compute the least fixed point of $\gamma^2_{\mathcal{KB}}$. Beginning with $J_0 = \emptyset$, we compute $s\mathcal{P}^{J_0}_{\Sigma}$. This program contains (r_1), (r_2) and (r_3) together with all ground instances of (r_4) and, for every t, the rule

$$\mathsf{oneDayTrip}(t) \leftarrow DL[\mathsf{Destination} \uplus \mathsf{wineDest}; \mathsf{Destination}](t)\,.$$

The least model of $\mathcal{KB}^{J_0}$ is

$$\begin{aligned} J_1 = {} & \{\mathsf{wineDest}(\mathsf{Tasmania}), \mathsf{wineDest}(\mathsf{Sydney}), \mathsf{wineDest}(\mathsf{TamarValley})\} \cup \\ & \cup \{\mathsf{oneDayTrip}(\mathsf{Tasmania}), \mathsf{oneDayTrip}(\mathsf{TamarValley})\} \cup \\ & \cup \{\mathsf{oneDayTrip}(t) \mid \Sigma \models \mathsf{Destination}(t)\} \cup \{\mathsf{overnight}(\mathsf{Sydney})\} \end{aligned}$$

Now we compute $s\mathcal{P}^{J_1}_{\Sigma}$. As discussed in Example 4, since the reduct of $\mathcal{KB}$ only depends on the instances of overnight *included in the interpretation, we have that $s\mathcal{P}^{J_1}_{\Sigma} = s\mathcal{P}^{I_2}_{\Sigma}$ with I_2 as before. The least model of $\mathcal{KB}^{J_1}$ is therefore $J_2 = I_2$, and this coincides with $\gamma^2_{\mathcal{KB}}(J_0)$.*

Continuing this process, we compute $s\mathcal{P}^{J_2}_{\Sigma}$, which is again $s\mathcal{P}^{I_2}_{\Sigma}$, and conclude that its least model is $J_3 = J_2 = I_2$. Therefore $I_2 = \mathsf{lfp}\left(\gamma_{\mathcal{KB}}\right)$; hence, $I_2 = \gamma_{\mathcal{KB}}(I_2) = \gamma^2_{\mathcal{KB}}(I_2)$ is also the least fixed point of $\gamma^2_{\mathcal{KB}}$.

One can also verify that the greatest fixed point of $\gamma^2_{\mathcal{KB}}$ coincides again with I_2. Therefore, the well-founded semantics of $\mathcal{KB}$ contains all atoms in I_2 together with the negations of all other closed atoms in $\mathsf{HB}_{\mathcal{KB}}$. In this case, negation is acyclic, which implies that all closed atoms are either well-founded or unfounded, so in particular there is only one answer set (as mentioned earlier), and the positive part of this well-founded semantics (I_2) is a model of $\mathcal{KB}$ – see Theorem 5.9 of [17] for details.

2.4. Syntax of multi-context systems

We now turn to multi-context systems, summarizing their syntax and different semantics. As before, we use a running example to illustrate these concepts.

The notion of multi-context system is defined in several layers. We summarize the definitions from [4].

Definition 1.

1. *A* logic *is a triple $\Sigma = \langle KB, BS, ACC \rangle$ where KB is the set of well-formed knowledge bases of Σ; BS is the set of possible belief sets; and $ACC : KB \to 2^{BS}$ is a function assigning to each element of KB a set of acceptable sets of beliefs.*

2. *Given a set of logics $\{\Sigma_1, \ldots, \Sigma_n\}$, where each $\Sigma_i = \langle KB_i, BS_i, ACC_i \rangle$, a* Σ_k-bridge rule, *with $1 \leq k \leq n$, has the form*

$$s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), \mathbf{not}(r_{j+1} : p_{j+1}), \ldots, \mathbf{not}(r_m : p_m)$$

   where $1 \leq r_\ell \leq n$; p_ℓ is an element of some belief set of Σ_{r_ℓ}; and $kb \cup \{s\} \in KB_k$ for each $kb \in KB_k$.

3. *A* multi-context system *(MCS) $M = \langle C_1, \ldots, C_n \rangle$ is a collection of contexts $C_i = \langle \Sigma_i, kb_i, br_i \rangle$, where $\Sigma_i = \langle KB_i, BS_i, ACC_i \rangle$ is a logic, $kb_i \in KB_i$ is a knowledge base, and br_i is a set of Σ_i-bridge rules over $\{\Sigma_1, \ldots, \Sigma_n\}$.*

In order to make these notions as general as possible, the definition of logic does not make any assumptions about what exactly knowledge bases or belief sets are. The former are part of the syntax of the language, their precise definition being left to Σ, while the latter intuitively represent the sets of syntactical elements representing the beliefs an agent may adopt. Likewise, ACC is an abstract description of the semantics of the logic.

Example 6. *We introduce a running example of multi-context system that we use throughout the remainder of this section. It is an MCS $M = \langle C_1, C_2, C_3\rangle$ defined as follows, where $\mathcal{A}$ is a set of constants (including at least* john *and* jack*) that we assume fixed.*

- *C_1 is a database context with a single unary relation* student*. Its underlying logic is defined by $KB_1 = BS_1 = \wp\{\mathsf{student}(t) \mid t \in \mathcal{A}\}$, where $\wp$ denotes the powerset operator, and $ACC_1(kb) = \{kb\}$. Furthermore, $kb_1 = \{\mathsf{student}(\mathsf{john}), \mathsf{student}(\mathsf{jack})\}$ and $br_1 = \emptyset$.*

- *C_2 is an ontology with a single axiom $\mathsf{Student} \sqsubseteq \mathsf{Person}$ and empty A-Box; intuitively, the A-Box will be filled from C_1's knowledge base by means of bridge rules. We assume a very simple description logic with no concept or role constructs and no negation, so its set of formulas is simply $F = \{\mathsf{Student}(t), \mathsf{Person}(t), t = t' \mid t, t' \in \mathcal{A}\} \cup \{\mathsf{Student} \sqsubseteq \mathsf{Person}, \mathsf{Person} \sqsubseteq \mathsf{Student}\}$.*

  Its underlying logic has $KB_2 = BS_2 = \wp(F)$ and $ACC_2(kb) = \{\mathsf{Th}(kb)\}$, where $\mathsf{Th}(kb)$ contains all formulas entailed by kb in the given description logic. Furthermore, $kb_2 = \{\mathsf{Student} \sqsubseteq \mathsf{Person}\}$ and $br_2 = \{(2 : \mathsf{Student}(t)) \leftarrow (1 : \mathsf{student}(t)) \mid t \in \mathcal{A}\}$.

  For readibility and succinctness, it is customary to write bridge rules with variables as a shorthand notation for all their ground instances [5]. We adopt this practice in this presentation; thus, the above definition of br_2 can be simplified to $br_2 = \{(2 : \mathsf{Student}(X)) \leftarrow (1 : \mathsf{student}(X))\}$.

- *C_3 is a view context with a binary relation* tutor*, with intended meaning that every student should have either* anne *or* kate *as a tutor. Its underlying logic is defined similarly to that of C_1: $KB_3 = BS_3 = \wp\{\mathsf{tutor}(t_1, t_2) \mid t_1 \in \mathcal{A}, t_2 \in \{\mathsf{anne}, \mathsf{kate}\}\}$ and $ACC_3(kb) = \{kb\}$.*

  Furthermore, $kb_3 = \emptyset$ and br_3 contains the two rules

$$(3 : \mathsf{tutor}(X, \mathsf{anne})) \leftarrow (1 : \mathsf{student}(X)), \mathbf{not}(3 : \mathsf{tutor}(X, \mathsf{kate})) \qquad (r_6)$$

$$(3 : \mathsf{tutor}(X, \mathsf{kate})) \leftarrow (1 : \mathsf{student}(X)), \mathbf{not}(3 : \mathsf{tutor}(X, \mathsf{anne})) \qquad (r_7)$$

2.5. Semantics of multi-context systems

The semantics for multi-context systems are defined in terms of the semantics for the logics in the individual contexts.

Definition 2. *Let $M = \langle C_1, \ldots, C_n\rangle$ be a multi-context system, where, for each $1 \leq i \leq n$, $C_i = \langle \Sigma_i, kb_i, br_i\rangle$.*

1. *A* belief state *for M is a collection $S = \langle S_1, \ldots, S_n\rangle$ of belief sets for each context, i.e. $S_i \in BS_i$ for each $1 \leq i \leq n$.*
2. *A bridge rule $s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), \mathbf{not}(r_{j+1} : p_{j+1}), \ldots, \mathbf{not}(r_m : p_m)$ is* applicable *in a belief state $S = \langle S_1, \ldots, S_n\rangle$ iff $p_i \in S_{r_i}$ for $1 \leq i \leq j$ and $p_k \notin S_{r_k}$ for $j + 1 \leq k \leq m$.*
3. *A belief state $S = \langle S_1, \ldots, S_n\rangle$ of M is an* equilibrium *iff the condition $S_i \in ACC_i(kb_i \cup \{H(r) \mid r \in br_i$ is applicable in $S\})$ holds for $1 \leq i \leq n$, where $H(r)$ denotes the head of rule r as usual.*

Belief states are "candidate" models, in the sense that they are well-formed potential models of each context. Information is transported between different contexts by means of bridge rules, and the notion of equilibrium guarantees that all belief states not only are models of the local information at each context, but also reflect the relationships imposed by the bridge rules. These are interpreted as logical implications: if a bridge rule is applicable in an equilibrium, then the information in its head must be included in the corresponding belief set, but if that rule is not applicable then this information may or may not be included.

Example 7. *In the setting of Example 6, consider the belief states* $S_a = \langle S^1, S^2, S^3_a \rangle$, $S_b = \langle S^1, S^2, S^3_b \rangle$ *and* $S_c = \langle S^1, S^2, S^3_c \rangle$ *with:*

$$\begin{aligned} S^1 = kb_1 &= \{\mathsf{student(john)}, \mathsf{student(jack)}\} \\ S^2 &= \{\mathsf{Student(john)}, \mathsf{Student(jack)}, \mathsf{Person(john)}, \mathsf{Person(jack)}, \mathsf{Student} \sqsubseteq \mathsf{Person}\} \\ S^3_a &= \{\mathsf{tutor(jack, anne)}\} \\ S^3_b &= \{\mathsf{tutor(jack, anne)}, \mathsf{tutor(john, anne)}\} \\ S^3_c &= \{\mathsf{tutor(jack, anne)}, \mathsf{tutor(john, anne)}, \mathsf{tutor(jack, kate)}\} \end{aligned}$$

Here, S_b *is an equilibrium for* M*, but* S_a *and* S_c *are not. In the case of* S_b*, the heads of the applicable bridge rules in* br_3 *give exactly the atoms already in* S^3_b*, and* $S^3_b \in ACC_3\left(S^3_b\right)$*. In the case of* S_a*, both* (r_6) *and* (r_7) *are applicable with* $X = \mathsf{john}$*, and* (r_6) *is also applicable with* $X = \mathsf{jack}$*; however,* $S^3_a \notin ACC_3\left(\{\mathsf{tutor(jack, anne)}, \mathsf{tutor(john, anne)}, \mathsf{tutor(john, kate)}\}\right)$*. In the case of* S_c*, none of the rules is applicable with* $X = \mathsf{jack}$*,* (r_6) *is applicable with* $X = \mathsf{john}$*, and again* $S^3_c \notin ACC_3(\{\mathsf{tutor(john, anne)}\})$*.*

Minimal and grounded equilibria. As in logic programming, there are often too many equilibria for a given multi-context system. For this reason, several particular kinds of equilibria are defined in [4], reflecting different kinds of preferences one may adopt to choose among them. These categories closely follow the usual hierarchy for models of logic programs, as well as of dl-programs.

The basic concept is that of minimal equilibrium. This is a relative notion, since (as discussed in [4]) it may not make sense to minimize the belief sets for *all* contexts.

Definition 3. *Let* $M = \langle C_1, \ldots, C_n \rangle$ *be a multi-context system and* $C^* \subseteq \{C_1, \ldots, C_n\}$ *be the set of contexts of* M *whose models should be minimized. An equilibrium* $S = \langle S_1, \ldots, S_n \rangle$ *of* M *is* C^*-minimal *if there is no equilibrium* $S' = \langle S'_1, \ldots, S'_n \rangle$ *of* M *such that:*

1. $S'_i \subseteq S_i$ *for all* $C_i \in C^*$*;*
2. $S'_i \subsetneq S_i$ *for some* $C_i \in C^*$*;*
3. $S'_i = S_i$ *for all* $C_i \notin C^*$*.*

In most examples hereafter, we use $C^* = M$; in this case, we simply refer to *minimal* equilibria, which the reader should understand to mean M-minimal equilibria.

Example 8. *In Example 7,* S_b *is a minimal equilibrium: both* S^1 *and* S^2 *are the unique possible belief states for* C_1 *and* C_2 *in an equilibrium, and the bridge rules in* br_3 *imply that no subset of* S^3_2 *can be part of an equilibrium (by a reasoning similar to the one showing that* S_a *is not an equilibrium).*

Other minimal equilibria for M *exist, namely corresponding to all belief states for* C_3 *in which both* john *and* jack *have exactly one tutor.*

Minimal equilibria (or even C^*-equilibria) do not necessarily exist. In logic programming, it is shown that least models always exist for positive programs, a result that holds also for dl-programs [17]. In MCSs, this class corresponds to that of definite multi-context systems.

Definition 4. *1. A logic* Σ *is* monotonic *if* $ACC_\Sigma(kb)$ *is always a singleton set, and* $kb \subseteq kb'$ *implies that the only element of* $ACC_\Sigma(kb)$ *is a subset of the only element of* $ACC_\Sigma(kb')$*. This coincides with the usual notion of monotonic logic.*

2. *A logic* $\Sigma = \langle KB_\Sigma, BS_\Sigma, ACC_\Sigma \rangle$ *is* reducible *if*
   (a) *there is* $KB^*_\Sigma \subseteq KB_\Sigma$ *such that the restriction of* Σ *to* KB^*_Σ *is monotonic;*
   (b) *there is a reduction function* $\mathsf{red}_\Sigma : KB_\Sigma \times BS_\Sigma \to KB^*_\Sigma$ *such that for each* $k \in KB_\Sigma$ *and* $S, S' \in BS_\Sigma$*:*

- $\mathsf{red}_\Sigma(k, S) = k$ *whenever* $k \in KB^*_\Sigma$*;*
- red_Σ *is anti-monotonic in the second argument, that is,* $\mathsf{red}_\Sigma(k, S) \subseteq \mathsf{red}_\Sigma(k, S')$ *whenever* $S' \subseteq S$*, where* $S' \subseteq S$ *holds if* $S = \langle S_1, \ldots, S_n\rangle$*,* $S' = \langle S'_1, \ldots, S'_n\rangle$ *and* $S'_i \subseteq S_i$ *for all* i*;*
- $S \in ACC_\Sigma(k)$ *iff* $ACC_\Sigma(\mathsf{red}_\Sigma(k, S)) = \{S\}$.

3. *A context* $C = (\Sigma, kb, br)$ *is* reducible *if*

   (a) Σ *is reducible;*

   (b) *for all* $H \subseteq \{H(r) \mid r \in br\}$ *and belief sets* S, $\mathsf{red}_\Sigma(kb \cup H, S) = \mathsf{red}_\Sigma(kb, S) \cup H$.

4. *A multi-context system is* reducible *if all of its contexts are reducible.*

Example 9. *All contexts in Example 6 are monotonic, and therefore trivially reducible with* $KB^* = KB$.

The proof of Lemma 1, in Section 3.3, includes a more interesting example of a reducible MCS.

A *definite* MCS is a reducible MCS in which bridge rules are monotonic (that is, they do not contain **not**) and knowledge bases are in reduced form (that is, $kb_i = red_i(kb_i, S)$ for all i and every $S \in BS_i$). Every definite MCS has a unique minimal equilibrium [4], which we denote by $Eq(M)$. The semantics of non-definite MCSs is defined via a generalization of the Gelfond–Lifschitz reduct to the multi-context case.

Definition 5. *Let* $M = \langle C_1, \ldots, C_n\rangle$ *be a reducible MCS, where context* C_i *has a reduction function* red_{Σ_i}*, and let* $S = \langle S_1, \ldots, S_n\rangle$ *be a belief state of* M.

The S-reduct *of* M *is* $M^S = \langle C_1^S, \ldots, C_n^S\rangle$*, where* $C_i^S = \langle \Sigma_i, \mathsf{red}_{\Sigma_i}(kb_i, S_i), br_i^S\rangle$ *and, for each* i*,* br_i^S *is obtained from* br_i *by (1) deleting every rule with some* **not** $(k : p)$ *in the body such that* $p \in S_k$*, and (2) deleting all* **not** *literals from the bodies of remaining rules.*

If $S = Eq(M^S)$*, then* S *is a* grounded equilibrium *of* M.

This definition only makes sense if M^S is definite. It has been shown [4] that this is always the case. In particular, if M is a definite MCS, then its minimal equilibrium is its only grounded equilibrium. In other cases, several grounded equilibria (or none) may exist. It is also easy to verify that grounded equilibria of M are indeed equilibria of M.

Example 10. *Consider again the MCS* M *from Example 6 and equilibrium* S_a *from Example 7. We assume* $\mathcal{A} = \{\mathsf{john}, \mathsf{jack}\}$*. Since in this case* $KB_i^* = KB_i$*, the first condition imposed on reduction functions requires that they all be projections:* $\mathsf{red}_i(k, S) = k$ *for all* $k \in KB_i$*. As a consequence,* M *and* M^{S_a} *can only differ in the sets of bridge rules in their contexts. Furthermore, the bridge rules in* C_1 *and* C_2 *do not have negations in their bodies, so* $br_1^{S_a} = br_1$ *and* $br_2^{S_a} = br_2$.

The case of $br_3^{S_a}$ *is more interesting. For rule* (r_6)*, we always have that* $\mathsf{tutor}(X, \mathsf{kate}) \notin S_a^3$*, so by condition (2) of the above definition we keep both instances of this rule, but delete their negative literal. For rule* (r_7)*, since* $\mathsf{tutor}(\mathsf{jack}, \mathsf{anne}) \in S_a^3$*, we remove the instance where* $X = \mathsf{jack}$*, by condition (1), and keep the instance where* $X = \mathsf{john}$*, removing its negative literal. Thus* $br_3^{S_a}$ *contains*

$$(3 : \mathsf{tutor}(X, \mathsf{anne})) \leftarrow (1 : \mathsf{student}(X)) \qquad \textit{and} \qquad (3 : \mathsf{tutor}(\mathsf{john}, \mathsf{kate})) \leftarrow (1 : \mathsf{student}(\mathsf{john}))$$

Well-founded semantics. The well-founded semantics for reducible MCSs is also defined in [4]. As usual, this semantics is based on the operator $\gamma_M(S) = Eq(M^S)$. Since γ_M is anti-monotonic, γ_M^2 is monotonic as usual. However, one can only guarantee the existence of its least fixpoint by the Knaster–Tarski theorem if BS_i has a least element for each logic Σ_i in any of M's contexts. If this is the case, then the well-founded semantics of M is $WFS(M) = \mathsf{lfp}\left(\gamma_M^2\right)$.

As in logic programming (and dl-programs), $WFS(M)$ is not necessarily an equilibrium: informally, it contains the knowledge that is common to all grounded equilibria, but being an equilibrium is not preserved by intersection.

Example 11. *Consider the multi-context system M from Example 6. As discussed in Example 10, S-reducts of M only differ from M in br_3. Starting with $S_0 = \langle \emptyset, \emptyset, \emptyset \rangle$, we keep all rules but delete their negative parts, so that $br_3^{S_0}$ contains*

$$(3 : \mathsf{tutor}(t, \mathsf{anne})) \leftarrow (1 : \mathsf{student}(t)) \text{ and } (3 : \mathsf{tutor}(t, \mathsf{kate})) \leftarrow (1 : \mathsf{student}(t)) \text{ for all } t \in \mathcal{A}$$

in br_3. The minimal equilibrium for this MCS is

$$Eq\left(M^{S_0}\right) = S_1 = \langle S^1, S^2, \{\mathsf{tutor}(\mathsf{john}, \mathsf{anne}), \mathsf{tutor}(\mathsf{john}, \mathsf{kate}), \mathsf{tutor}(\mathsf{jack}, \mathsf{kate}), \mathsf{tutor}(\mathsf{jack}, \mathsf{anne})\}\rangle\,.$$

Now we compute M^{S_1}. We have to remove from br_3 all rules involving john *or* jack*, as the negated atom in their bodies is always in S_1's third component; thus, $br_3^{S_1}$ becomes*

$$(3 : \mathsf{tutor}(t, \mathsf{anne})) \leftarrow (1 : \mathsf{student}(t)) \text{ and } (3 : \mathsf{tutor}(t, \mathsf{kate})) \leftarrow (1 : \mathsf{student}(t)) \text{ for all } t \in \mathcal{A} \setminus \{\mathsf{john}, \mathsf{jack}\}$$

and $Eq\left(M^{S_1}\right) = \gamma_M^2(S_0) = S_2 = \langle S^1, S^2, \emptyset \rangle$.

Iterating this construction is very similar: $M^{S_2} = M^{S_0}$ and $Eq\left(M^{S_2}\right) = S_3$ is similar to S_1, but containing S^2 in its second component. Then $M^{S_3} = M^{S_1}$ and $Eq\left(M^{S_3}\right) = S_2 = \mathsf{lfp}\left(\gamma_M\right)$. Thus $S_2 = \langle S^1, S^2, \emptyset \rangle$ is the well-founded semantics of M. Observe that this belief state is not an equilibrium of M.

3. From dl-programs to multi-context systems

In this section, we define the (syntactic) translation from dl-programs to MCSs. Then, we prove the correspondence results that relate the semantics of a dl-program and the corresponding MCS. We illustrate this construction by applying it to the dl-program from Example 2.

The converse translation is trivially not possible – MCSs are far more general than dl-programs, since they are prepared to handle any kind of reasoning system.

3.1. The syntactic translation

There are two essential differences between dl-programs and MCSs: the former only allow the combination of a logic program with a description logic knowledge base, whereas the latter allow any kind of different systems to be joined together; and the former allow for *local* changes to the knowledge base, via input contexts in dl-atoms, whereas the interaction within the latter is *global*, since inferences derived from bridge rules can be used in further inferences in other contexts.

A third aspect concerns the way dl-programs and MCSs interact with the outside world. In both cases, we are faced with several independent components connected by a set of rules – a logic program, in the case of dl-programs, and the sets of bridge rules, in multi-context systems. However, all components of an MCS are observable from the outside world, since models of MCSs contain models of each individual component, whereas interaction with dl-programs must always be made via its logic program.

Because of these differences, the two systems have different kinds of expressiveness. In spite of this, it is simple to translate a dl-program into an MCS, as we now show. Given a dl-program $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$, there are two steps in the process of generating an MCS from $\mathcal{KB}$.

1. We split $\mathcal{P}$ into its purely logical part and its communication part, translating rules that contain dl-atoms into bridge rules.

2. For each distinct input context χ appearing in $\mathcal{P}$, we create a different copy of the knowledge base, corresponding to the view of the knowledge base within the dl-atoms containing χ.

Although the idea behind this syntactic construction is suggested in [5], it is not defined precisely therein, neither are its semantic implications discussed – so even though the authors use it to justify that MCSs generalize dl-programs, this claim is not substantiated beyond an intuitive perspective. Here, we not only make this syntactic correspondence precise, but discuss in detail the semantic correspondences it entails.

The first step is to encapsulate a description logic knowledge base in a context.

Definition 6. *A description logic $\mathcal{L}$ induces the logic $L(\mathcal{L}) = \langle KB_\mathcal{L}, BS_\mathcal{L}, ACC_\mathcal{L}\rangle$, where $KB_\mathcal{L}$ is the set of all well-formed knowledge bases of $\mathcal{L}$; $BS_\mathcal{L}$ is the set of all sets of dl-queries (defined as for dl-programs) in the language of $\mathcal{L}$; and $ACC_\mathcal{L}(kb)$ is the singleton set containing the set of queries $Q(\bar{t})$ such that $kb \models Q(\bar{t})$, also denoted $kb^{\models}$.*

The context induced by a knowledge base Σ over $\mathcal{L}$ is $\mathsf{Ctx}(\Sigma) = \langle L(\mathcal{L}), \Sigma, \emptyset\rangle$.

Since description logics have open-world semantics, a query such as $\neg\mathsf{C}(\mathsf{a})$ is only in $kb^{\models}$ if $\neg\mathsf{C}(\mathsf{a})$ can be proved from kb (and not, as in databases, simply if $\mathsf{C}(\mathsf{a})$ cannot be proved). As a consequence, the belief sets (the elements of $BS_\mathcal{L}$) need not be categorical: they may contain neither $\mathsf{C}(\mathsf{a})$ nor $\neg\mathsf{C}(\mathsf{a})$ for a particular concept C and individual a. Also, since description logics are monotonic, if $kb \subseteq kb'$, then $kb^{\models} \subseteq kb'^{\models}$.

Definition 7. *Let $\mathcal{KB} = \langle \Sigma, \mathcal{P}\rangle$ be a dl-program and $\chi_1, \ldots, \chi_n$ be the distinct input contexts occurring in dl-atoms in $\mathcal{P}$.*

1. *The translation $\sigma_\mathcal{KB}$ of literals and dl-atoms is defined as follows:*

$$\sigma_\mathcal{KB}(a) = \begin{cases} (0 : a) & \text{if } a \text{ is an atom} \\ \mathbf{not}(0 : b) & a = \mathbf{not}\, b \text{ for some atom } b \\ (i : Q(\bar{t})) & a = DL[\chi_i; Q](\bar{t}) \\ \mathbf{not}(i : Q(\bar{t})) & a = \mathbf{not}\, DL[\chi_i; Q](\bar{t}) \end{cases}$$

2. *The translation of $\mathcal{P}$ is the context $C_0 = \langle L_0, kb_0, br_0\rangle$ where:*
   - *$L_0 = \langle KB_0, BS_0, ACC_0\rangle$ is the logic underlying $\mathcal{P}$, where KB_0 is the set of all logic programs over $\mathcal{P}$'s signature, BS_0 is the power set of $\mathsf{HB}_\mathcal{P}$, and ACC_0 assigns each program to the set of its first-order models (restricted to $\mathsf{HB}_\mathcal{P}$);*
   - *kb_0 is $\mathcal{P}^-$, the set of rules of $\mathcal{P}$ that do not contain any dl-atoms;*
   - *br_0 contains $p \leftarrow \sigma_\mathcal{KB}(l_1), \ldots, \sigma_\mathcal{KB}(l_m)$ for each rule $p \leftarrow l_1, \ldots, l_m$ in $\mathcal{P} \setminus \mathcal{P}^-$.*
3. *Each input context $\chi_i = P_1\ op_1\ p_1, \ldots, P_k\ op_k\ p_k$, with $i = 1, \ldots, n$, generates a context $C_i = \langle L(\mathcal{L}), \Sigma, br_i\rangle$ where br_i contains all ground instances of the bridge rules $P_j(X) \leftarrow (0 : p_j(X))$, if P_j is a concept and $op_j = \uplus$, or $\neg P_j(X) \leftarrow (0 : p_j(X))$, if P_j is a concept and $op_j = \cup$, or $P_j(X,Y) \leftarrow (0 : p_j(X,Y))$, if P_j is a role and $op_j = \uplus$, or $\neg P_j(X,Y) \leftarrow (0 : p_j(X,Y))$, if P_j is a role and $op_j = \cup$, for $j = 1, \ldots, k$.*
4. *The multi-context system* generated *by $\mathcal{KB}$ is $\mathsf{M}(\mathcal{KB}) = \langle C_0, C_1, \ldots, C_n\rangle$.*

Recall that bridge rules with variables are shorthand notation for all their ground instances (see Example 6).

In Section 4.1 we describe how this translation can be extended to the full language of dl-programs including $\curlywedge$.

The first context in $\mathsf{M}(\mathcal{KB})$ is a logic program with the same underlying language of $\mathcal{P}$. This implies that any interpretation I of $\mathcal{P}$ is an element of BS_0, and vice-versa. We use this fact hereafter without mention.

Example 12. *Recall the dl-program $\mathcal{KB}$ from Example 2. For the purpose of generating an MCS from $\mathcal{KB}$, observe that there are two different input contexts in this program, $\chi_1 = \epsilon$ and $\chi_2 = \mathsf{Destination} \uplus \mathsf{wineDest}$. Rules (r_1), (r_2) and (r_3) do not include dl-atoms, so they belong to $\mathcal{P}^-$.*

Rules (r_4) and (r_5), which contain dl-atoms, are translated as the following bridge rules.

$$\mathsf{overnight}(X) \leftarrow (1 : \mathsf{hasAccommodation}(X, Y)) \qquad (r'_4)$$

$$\mathsf{oneDayTrip}(X) \leftarrow (2 : \mathsf{Destination}(X)), \mathbf{not}(0 : \mathsf{overnight}(X)) \qquad (r'_5)$$

The generated multi-context system $\mathsf{M}(\mathcal{KB})$ is thus $\langle C_0, C_1, C_2\rangle$, where:

- $C_0 = \langle L_0, \{r_1, r_2, r_3\}, \{r'_4, r'_5\}\rangle$
- $C_1 = \langle L, \Sigma, \emptyset\rangle$
- $C_2 = \langle L, \Sigma, \{\mathsf{Destination}(X) \leftarrow (0 : \mathsf{wineDest}(X))\}\rangle$

As stated earlier, $\mathsf{M}(\mathcal{KB})$ *actually contains the ground versions of the bridge rules given here.*

3.2. Relating the semantics

Just as we can generate a multi-context system $\mathsf{M}(\mathcal{KB})$ from any dl-program $\mathcal{KB}$, we can generate a belief state for $\mathsf{M}(\mathcal{KB})$ from any interpretation of $\mathcal{KB}$.

Definition 8. *Let* $\mathcal{KB} = \langle \Sigma, \mathcal{P}\rangle$ *be a dl-program and* I *be an interpretation of* $\mathcal{KB}$*. The belief state* generated *by* I *is* $\mathsf{S}_{\mathcal{KB}}(I) = \langle S_0^I, S_1^I, \ldots, S_n^I\rangle$ *of* $\mathsf{M}(\mathcal{KB})$*, where* $S_0^I = I$ *and, for* $i = 1, \ldots, n$*,* S_i^I *is*

$$\left(\Sigma \cup \{P(\bar{t}) \mid I \models p(\bar{t}), P \uplus p \in \chi_i\} \cup \{\neg P(\bar{t}) \mid I \models p(\bar{t}), P \uplus\ p \in \chi_i\}\right)^{\models}.$$

Intuitively, S_i^I simply corresponds to the view of the knowledge base by extending Σ with the information from the input context χ_i.

It is straightforward to verify that $\mathsf{S}_{\mathcal{KB}}(I)$ is a belief state of $\mathsf{M}(\mathcal{KB})$. When there is only one dl-program under consideration, we omit the subscript in $\mathsf{S}_{\mathcal{KB}}$.

Example 13. *Recall the interpretations for the dl-program* $\mathcal{KB}$ *from Example 3. The belief states generated by these interpretations all contain the interpretation itself as the belief set for* C_0 *and* $\Sigma^{\models}$ *as belief set for* C_1 *(since* $\chi_1 = \emptyset$ *and this is the knowledge base corresponding to* Σ*). Furthermore,* $\Sigma^{\models}$ *is a subset of the belief set for* C_2 *in all cases, by monotonicity of* Σ*'s underlying logic.*

Interpretation I_1 *satisfies* $\mathsf{wineDest}(\mathsf{Tasmania})$*,* $\mathsf{wineDest}(\mathsf{Sydney})$ *and* $\mathsf{wineDest}(\mathsf{TamarValley})$*, whence* $\mathsf{Destination}(\mathsf{Tasmania})$*,* $\mathsf{Destination}(\mathsf{Sydney})$ *and* $\mathsf{Destination}(\mathsf{TamarValley})$ *are included in the belief set for* C_2 *–* $\mathsf{Destination}(\mathsf{Sydney})$ *was actually already there. Since there are no axioms involving* $\mathsf{Destination}$ *in* Σ*, these are the only new elements of this belief set. Thus:*

$$\mathsf{S}(I_1) = \langle I_1, kb^{\models}, kb^{\models} \cup \{\mathsf{Destination}(\mathsf{Tasmania}), \mathsf{Destination}(\mathsf{TamarValley})\}\rangle$$

The case of I_2 *is similar, and*

$$\mathsf{S}(I_2) = \langle I_2, kb^{\models}, kb^{\models} \cup \{\mathsf{Destination}(\mathsf{Tasmania}), \mathsf{Destination}(\mathsf{TamarValley})\}\rangle$$

In the previous example, one can check that $\mathsf{S}(I_2)$ is the only belief state that is also an equilibrium of $\mathsf{M}(\mathcal{KB})$, and I_2 was also the only model of $\mathcal{KB}$. This suggests that there are very close connections between I and $\mathsf{S}(I)$, which we now formally state and prove.

Lemma 1. *Let* $\mathcal{KB} = \langle \Sigma, \mathcal{P}\rangle$ *be a dl-program and* $DL[\chi_i; Q](\bar{t})$ *be a ground dl-atom in* $\mathcal{P}$*.*

1. *For any interpretation* I*,* $I \models_\Sigma DL[\chi_i; Q](\bar{t})$ *iff* $Q(\bar{t}) \in S_i^I$*.*
2. *If* $S = \langle S_0, S_1, \ldots, S_n\rangle$ *is an equilibrium of* $\mathsf{M}(\mathcal{KB})$ *and* $1 \leq i \leq n$*, then* $Q(\bar{t}) \in S_i$ *iff* $S_0 \models_\Sigma DL[\chi_i; Q](\bar{t})$*.*

Proof. The first equivalence is straightforward, since the construction of S_i^I mimics the definition of the semantics of $\mathcal{KB}$.

For the second equivalence, $S_0 \models_\Sigma DL[\chi_i; Q](\bar{t})$ holds iff $\Sigma(S_0; \chi_i) \models Q(\bar{t})$ iff Σ, extended with every instance $P(\bar{e})$ such that $p(\bar{e}) \in S_0$, if $P \uplus p \in \chi_i$, and every instance $\neg P(\bar{e})$ such that $p(\bar{e}) \notin S_0$, if $P \uplus\ p \in \chi_i$, satisfies $Q(\bar{t})$. This last condition is equivalent to

$$Q(\bar{t}) \in \left(\Sigma \cup \{P(\bar{t}) \mid I \models p(\bar{t}), P \uplus p \in \chi_i\} \cup \{\neg P(\bar{t}) \mid I \models p(\bar{t}), P \uplus\ p \in \chi_i\}\right)^{\models},$$

which is the only element of ACC_i; since S is an equilibrium, this set concides with S_i. □

Theorem 1. *Let $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ be a dl-program.*

1. *If I is a model of $\mathcal{KB}$, then $\mathsf{S}(I)$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$.*
2. *If $S = \langle S_0, \ldots, S_n \rangle$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$, then S_0 is a model of $\mathcal{KB}$.*

Proof. 1. Suppose that I is a model of $\mathcal{KB}$ and let $\mathsf{S}(I) = \langle S_0^I, S_1^I, \ldots, S_n^I \rangle$ be the belief state generated by I. For each i, we need to show that S_i^I is an acceptable belief set for kb_i and that $s \in S_i^I$ whenever the bridge rule $s \leftarrow \sigma(l_1), \ldots, \sigma(l_k) \in br_i$ is applicable in $\mathsf{S}(I)$.

Consider first the case $i = 0$. Since $S_0^I = I$ is a model of all the rules in $\mathcal{P}$, it follows that I satisfies every rule in $kb_0 = \mathcal{P}^-$. Let $s \leftarrow \sigma_{\mathcal{KB}}(l_1), \ldots, \sigma_{\mathcal{KB}}(l_k)$ be a bridge rule in br_0; this must originate from a rule $s \leftarrow l_1, \ldots, l_k$ in $\mathcal{P} \setminus \mathcal{P}^-$. Assume that the bridge rule is applicable in $\mathsf{S}(I)$; for each l_j, there are three possibilities: (1) l_j is a regular literal, and then $I \models l_j$; (2) l_j is $DL[\chi_m; Q](\bar{t})$ and $Q(\bar{t}) \in S_m^I$, whence $I = S_0^I \models l_j$ by Lemma 1; (3) l_j is **not** $DL[\chi_m; Q](\bar{t})$ and $Q(\bar{t}) \notin S_m^I$, whence $I = S_0^I \not\models DL[\chi_m; Q](\bar{t})$ by Lemma 1, and therefore $I \models l_j$. Thus, I satisfies the body of the rule, hence $S_0^I = I \models s$.

Suppose now that $i \neq 0$. By construction, S_i^I is the only element of $ACC_i(\Sigma \cup \{P(\bar{t}) \mid I \models p(\bar{t}), P \uplus p \in \chi_i\} \cup \{\neg P(\bar{t}) \mid I \models p(\bar{t}), P \uplus\ p \in \chi_i\})$, which is precisely the belief set containing all queries $Q(\bar{t})$ that Σ, extended with the heads of the bridge rules applicable in $\mathsf{S}(I)$, satisfies.

Therefore $\mathsf{S}(I)$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$.

2. Suppose that $S = \langle S_0, S_1, \ldots, S_n \rangle$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$. Since S_0 is a model of kb_0 extended with the heads of bridge rules in br_0 which are applicable in S, it follows that S_0 satisfies all the rules of $kb_0 = \mathcal{P}^-$.

Let $p \leftarrow l_1, \ldots, l_k$ be a rule in $\mathcal{P} \setminus \mathcal{P}^-$. Then $p \leftarrow \sigma_{\mathcal{KB}}(l_1), \ldots, \sigma_{\mathcal{KB}}(l_k)$ is a bridge rule in br_0. Again, if S_0 satisfies the body of the rule, then the corresponding bridge rule is applicable in S: for regular literals this is immediate (the condition is the same), while for dl-atoms and their negations this is again Lemma 1. Hence S_0 is also a model of the remaining rules in $\mathcal{P}$.

Therefore S_0 is a model of $\mathcal{KB}$. □

In case 1 of the previous proof, nothing is said about the case when a bridge rule is not applicable: in this situation, the definition of equilibrium poses no restrictions on whether or not its head is part of the corresponding belief set. This can be seen e.g. in Example 13 by adding the literal $\mathsf{oneDayTrip}(\mathsf{Sydney})$ to I_2: this yields another model I_2' of $\mathcal{KB}$, and its translation is still an equilibrium for $\mathsf{M}(\mathcal{KB})$.

Furthermore, since ACC_i always yields a singleton set when $i \geq 1$, equilibria for MCSs generated from a dl-program can be uniquely derived from their first component, as expressed by the following corollary.

Corollary 1. *If $S = \langle S_0, \ldots, S_n \rangle$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$, then $\mathsf{S}(S_0) = S$.*

Proof. Suppose that S is an equilibrium of $\mathsf{M}(\mathcal{KB})$. In particular,

$$\begin{aligned} S_i &\in ACC_i(kb_i \cup \{H(r) \mid r \in br_i \text{ is applicable in } S\}) \\ &= ACC_i(\Sigma \cup \{P(\bar{t}) \mid S_0 \models p(\bar{t}), P \uplus p \in \chi_i\} \cup \{\neg P(\bar{t}) \mid S_0 \models p(\bar{t}), P \uplus\ p \in \chi_i\}) \end{aligned}$$

since $kb_i = \Sigma$, which is precisely the i-th component in $\mathsf{S}(S_0)$. □

This result allows us to state all future equivalences in terms of models of $\mathcal{P}$.

3.3. Answer-set semantics

Since the transformation S from interpretations of dl-programs to belief states preserves inclusions, we also have the following relationship.

Theorem 2. *Let $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ be a dl-program. Then I is a minimal model of $\mathcal{KB}$ iff $\mathsf{S}(I)$ is a minimal equilibrium of $\mathsf{M}(\mathcal{KB})$.*

Proof. ($\Rightarrow$) Suppose that I is a minimal model of $\mathcal{KB}$, and let $\mathsf{S}(I) = \langle S_0, S_1, \ldots, S_n\rangle$; by 1 in Theorem 1, $\mathsf{S}(I)$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$. Assume that there exists another equilibrium $S' = \langle S'_0, S'_1, \ldots, S'_n\rangle$ of $\mathsf{M}(\mathcal{KB})$ such that $S'_i \subseteq S_i$ for all i. In particular, $S'_0 \subseteq S_0 = I$, and by 2 in Theorem 1 S'_0 is a model of $\mathcal{KB}$. Since I is a minimal model of $\mathcal{KB}$ it follows that $S_0 = I$, whence by Corollary 1 $S' = \mathsf{S}(I)$. Therefore $\mathsf{S}(I)$ is a minimal equilibrium for $\mathsf{M}(\mathcal{KB})$.
($\Leftarrow$) Suppose that $\mathsf{S}(I) = \langle I, S_1, \ldots, S_n\rangle$ is a minimal equilibrium of $\mathsf{M}(\mathcal{KB})$. By 2 in Theorem 1, I is a model of $\mathcal{KB}$. Assume that there exists another model $I' \subseteq I$ of $\mathcal{KB}$. By 1 in Theorem 1 $\mathsf{S}(I') = \langle I', S'_1, \ldots, S'_n\rangle$ is an equilibrium of $\mathsf{M}(\mathcal{KB})$. Since $I' \subseteq I$, we also have that $S'_i \subseteq S_i$: every bridge rule from br_i that is applicable in $\mathsf{S}(I')$ is also applicable in $\mathsf{S}(I)$, so $(\Sigma \cup \{H(r) \mid r \in br_i \text{ is applicable in } \mathsf{S}(I')\}) \subseteq (\Sigma \cup \{H(r) \mid r \in br_i \text{ is applicable in } \mathsf{S}(I)\})$, and description logics are monotonic. Since $\mathsf{S}(I)$ is a minimal equilibrium for $\mathsf{M}(\mathcal{KB})$, this implies that $\mathsf{S}(I') = \mathsf{S}(I)$, and in particular $I = I'$. Therefore I is a minimal model of $\mathcal{KB}$. □

Example 14. *Recall the discussion after Theorem 1. Reconsider interpretations I_2 and I'_2, noting that $I_2 \subsetneq I'_2$. Clearly I'_2 is not a least model of $\mathcal{KB}$, whereas I_2 is. Correspondingly, $\mathsf{S}(I_2)$ is a minimal equilibrium of $\mathsf{M}(\mathcal{KB})$, but $\mathsf{S}(I'_2)$ is not: even though both belief states coincide on their second and third components, their first components are, respectively, I_2 and I'_2, and the former is a proper subset of the latter.*

We now show that answer sets for dl-programs correspond to grounded equilibria for MCSs. This is a reasonable result, since both the dl-transform of dl-programs and the reduct of an MCS are generalizations of the Gelfond–Lifschitz transform of ordinary logic programs.

Throughout the remainder of this section, let $\mathcal{KB} = \langle \Sigma, \mathcal{P}\rangle$ be a dl-program and $\mathsf{M}(\mathcal{KB})$ be the multi-context system generated by $\mathcal{KB}$, where $\mathsf{M}(\mathcal{KB}) = \langle C_0, C_1, \ldots, C_n\rangle$ and $C_i = \langle L_i, kb_i, br_i\rangle$ for $i = 0, \ldots, n$.

Proposition 1. *$\mathsf{M}(\mathcal{KB})$ is reducible, with KB^*_0 being the set of positive programs, $\mathsf{red}_0(\mathcal{P}, S) = \mathcal{P}^S$, computing the Gelfond–Lifschitz transform of $\mathcal{P}$ relative to S, and, for $i \geq 1$, $KB^*_i = KB_i$ and $\mathsf{red}_i(kb, S) = kb$ being a projection function.*

Proof. The logic L_0 is reducible, with KB^*_0 the set of positive programs and reduction function $\mathsf{red}_0(\mathcal{P}, S) = \mathcal{P}^S$, computing the Gelfond–Lifschitz transform.

The conditions for being a reducible logic are just well-known facts in logic programming, namely:

- positive logic programs are monotonic;
- the Gelfond–Lifschitz transform of a positive logic program is itself;
- if $S \subseteq S'$, then $\mathcal{P}^{S'} \subseteq \mathcal{P}^S$;
- S is a model of a (general) logic program $\mathcal{P}$ iff it is a model of $\mathcal{P}^S$.

Context C_0 is reducible: we need to show that $\mathsf{red}_0(\mathcal{P}^- \cup H, I) = \mathsf{red}_0(\mathcal{P}^-, I) \cup H$, for every interpretation I and any set H of heads of rules in br_0. But H consists solely of facts (rules with empty body), which are unaltered by the Gelfond–Lifschitz transform, hence this equality holds.

For $i = 1, \ldots, n$, context C_i is reducible via the identity function: since description logics are monotonic, we can take $KB^*_i = KB_i$, and the identity function trivially satisfies all the reducibility conditions. □

We now look closely at the relationship between the dl-transform of $\mathcal{P}$ and the reduction of $\mathsf{M}(\mathcal{KB})$. The former generates a subsystem of the latter in the following sense.

Definition 9. *Consider two multi-context systems $M = \langle C_1, \ldots, C_n\rangle$ and $M' = \langle C'_1, \ldots, C'_m\rangle$. We say that M' is a* subsystem *of M, $M' \subseteq M$, if there exists an injective function $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that C'_i is $C_{\varphi(i)}$ with every index j in $br_{\varphi(i)}$ replaced by $\varphi(j)$ for $1 \leq i \leq m$.*

In other words, a subsystem of M is a subset M' of M's contexts whose bridge rules do not refer to contexts outside M'. In the setting of Example 6, $\langle C_1, C_3\rangle$ is a subsystem of M, with φ the identity function. In particular, if $M' \subseteq M$, then $m \leq n$.

Let I be an interpretation of $\mathcal{KB}$, $M^{\mathsf{S}(I)} = \left\langle C_0^{\mathsf{S}(I)}, C_1^{\mathsf{S}(I)}, \ldots, C_n^{\mathsf{S}(I)} \right\rangle$ be the $\mathsf{S}(I)$-reduct of $\mathsf{M}(\mathcal{KB})$, and $\mathcal{KB}' = \langle \Sigma, sP_\Sigma^I \rangle$ be the dl-transform of $\mathcal{P}$ relative to Σ and I, with $M' = \mathsf{M}(\mathcal{KB}') = \langle C'_0, C'_1, \ldots, C'_m \rangle$ the corresponding multi-context system.

Proposition 2. *M' is a subsystem of $M^{\mathsf{S}(I)}$; also, it is a definite multi-context system.*

Proof. To see that M' is a subsystem of $M^{\mathsf{S}(I)}$, we first characterize M'. Since the set of input contexts χ'_i in sP_Σ^I is, in general, a subset of the χ_j in $\mathcal{P}$ (the removal of some dl-atoms and some dl-rules may have caused some input contexts to cease to occur), the indices in $\mathsf{M}(\mathcal{KB})$ and M' may not coincide. Let $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ be the appropriate renaming function, i.e. such that $\chi'_i = \chi_{\varphi(i)}$.

1. C'_0 is $C_0^{\mathsf{S}(I)}$, with every index $i \neq 0$ in br_0 replaced by $\varphi(i)$: this can be seen by observing that the rules removed from $\mathcal{P}$ in the construction of sP_Σ^I correspond precisely to the rules removed from C_0 in the construction of $C_0^{\mathsf{S}(I)}$. Consider a ground rule r obtained from grounding a rule in $\mathcal{P}$. If r does not contain dl-atoms, then $r \in sP_\Sigma^I$ iff $r \in (\mathcal{P}^-)^I = \mathsf{red}_0(kb_0, I)$.

   Suppose that r contains dl-atoms. Then r is not included in sP_Σ^I if r contains a negated atom or dl-atom l such that $I \not\models l$. If l is a negated atom $\neg p$, this means simply that $I \models p$, hence $p \in S_0^I = I$; if l is $\neg DL[\chi_i; Q](\bar{t})$, then by Lemma 1 $Q(\bar{t}) \in S_i^I$. In either case, the corresponding bridge rule is removed from br_0. This reasoning is reversible, so the converse implication also holds. If those conditions do not hold, then r is included in sP_Σ^I by removing its negated atoms or dl-atoms – and since $\sigma_{\mathcal{KB}}$ transforms these into **not** literals, the bridge rule obtained is the same as removing all **not** literals from the bridge rule derived from r.

2. $C'_i = C_{\varphi(i)}^{\mathsf{S}(I)}$: the only non-trivial part of this equality regards the bridge rules. But all bridge rules in C'_i are of the form $(\neg)P \leftarrow (0 : p)$, which do not contain negations in their body, so they are never removed.

This shows that M' is a subsystem of $M^{\mathsf{S}(I)}$. The fact that it is a definite MCS is a straightforward consequence of the definition of sP_Σ^I. □

From an interpretation J of $\mathcal{KB}$, we can generate belief states for M and M'; to distinguish them, we write the subscripts in S explicitly.

Proposition 3. *For any interpretation J of $\mathcal{KB}$, $\mathsf{S}_{\mathcal{KB}}(J)$ is a minimal equilibrium of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$ iff $\mathsf{S}_{\mathcal{KB}'}(J)$ is a minimal equilibrium of M'.*

Proof. The direct implication is straightforward. For the converse implication, note that from an equilibrium $S = \langle S_0, S_1, \ldots, S_n \rangle$ of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$ we can construct an equilibrium $S' = \langle S_0, S_{\varphi(1)}, \ldots, S_{\varphi(m)} \rangle$ of M'. Since M' is definite, it has a unique minimal equilibrium, and thus $\mathsf{S}_{\mathcal{KB}'}(J) \subseteq S'$, and in particular $J = {S'}_0^J \subseteq S_0$. Assume that $S \subseteq \mathsf{S}_{\mathcal{KB}}(J)$; then in particular $S_0 \subseteq S_0^J = J$, and necessarily $S_0 = J$, whence $S = \mathsf{S}_{\mathcal{KB}}(J)$. Thus $\mathsf{S}_{\mathcal{KB}}(J)$ is a minimal equilibrium for $\mathcal{KB}$. □

The following corollary, analogous to Corollary 1, will be essential later on.

Corollary 2. *Let $S = \langle S_0, S_1, \ldots, S_n \rangle$ be a minimal equilibrium of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$. Then $S = \mathsf{S}_{\mathcal{KB}}(S_0)$.*

Proof. Suppose $S = \langle S_0, S_1, \ldots, S_n \rangle$ is a minimal equilibrium of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$. From S, we can construct a minimal equilibrium $S' = \langle S_0, S_{\varphi(1)}, \ldots, S_{\varphi(m)} \rangle$ as in Proposition 3, which is a minimal equilibrium of M'. By Corollary 1, $S' = \mathsf{S}_{\mathcal{KB}'}(S_0)$, whence by Proposition 3, it follows that $\mathsf{S}_{\mathcal{KB}}(S_0)$ is also a minimal equilibrium of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$. But $M^{\mathsf{S}_{\mathcal{KB}}(I)}$ is a definite multi-context system, so it only has one minimal equililibrium. Hence $S = \mathsf{S}_{\mathcal{KB}}(S_0)$. □

Theorem 3. *I is an answer set for $\mathcal{KB}$ iff $\mathsf{S}(I)$ is a grounded equilibrium of $\mathsf{M}(\mathcal{KB})$.*

Proof. I is an answer set for $\mathcal{KB}$ iff I is the least model of $\mathcal{KB}'$ iff $\mathsf{S}_{\mathcal{KB}'}(I)$ is a minimal equilibrium of M' iff $\mathsf{S}_{\mathcal{KB}}(I)$ is a minimal equilibrium of $M^{\mathsf{S}_{\mathcal{KB}}(I)}$ iff $\mathsf{S}_{\mathcal{KB}}(I)$ is a grounded equilibrium of $\mathsf{M}(\mathcal{KB})$. □

3.4. Well-founded semantics

The definition of well-founded semantics for MCSs is very similar to that of well-founded semantics for dl-programs – in particular, it relies on a similarly defined anti-monotonic operator. We now make this correspondence precise. As before, let $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ be a dl-program, $M = \mathsf{M}(\mathcal{KB})$ be the MCS generated by $\mathcal{KB}$, and S be $\mathsf{S}_{\mathcal{KB}}$; observe that Proposition 3 guarantees that this makes sense in all formulas below.

Proposition 4. *For every interpretation I of $\mathcal{KB}$, $\gamma_M(\mathsf{S}(I)) = \mathsf{S}\left(\gamma_{\mathcal{KB}}(I)\right)$.*

Proof. Let I be an interpretation of $\mathcal{KB}$. By definition, $\gamma_M(\mathsf{S}(I))$ is the minimal equilibrium of $M^{\mathsf{S}(I)}$, the reduct of M relative to $\mathsf{S}(I)$, hence $\gamma_M(\mathsf{S}(I))$ may be written as $\mathsf{S}_{\mathcal{KB}}(J)$ for some J. By Proposition 3, $\mathsf{S}_{\mathcal{KB}'}(J)$ is a minimal equilibrium of $\mathsf{M}(\mathcal{KB}')$, with $\mathcal{KB}' = \langle \Sigma, sP^I_\Sigma \rangle$, and by Theorem 2 J is the least model of $\mathcal{KB}'$, hence $J = \gamma_{\mathcal{KB}}(I)$ as we wanted to show. □

Theorem 4. *I is the well-founded semantics of $\mathcal{KB}$ iff $\mathsf{S}(I)$ is the well-founded equilibrium of $\mathsf{M}(\mathcal{KB})$.*

Proof. The proof of this result amounts to showing that $\mathsf{lfp}\left(\gamma^2_M\right) = \mathsf{S}(\mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right))$. 1. For every interpretation I of $\mathcal{KB}$ and every $n > 0$, $\gamma^n_M(\mathsf{S}(I)) = \mathsf{S}\left(\gamma^n_{\mathcal{KB}}(I)\right)$.

For $n = 1$ this is simply Proposition 4. Assume the equality holds for n; then

$$\begin{aligned}\gamma^{n+1}_M(\mathsf{S}(I)) &= \gamma_M\left(\gamma^n_M(\mathsf{S}(I))\right) = \gamma_M\left(\mathsf{S}\left(\gamma^n_{\mathcal{KB}}(I)\right)\right) \\ &= \mathsf{S}\left(\gamma_{\mathcal{KB}}\left(\gamma^n_{\mathcal{KB}}(I)\right)\right) = \mathsf{S}\left(\gamma^{n+1}_{\mathcal{KB}}(I)\right) ,\end{aligned}$$

where the second equality holds by induction hypothesis and the third by Proposition 4.
2. Let $\bot = \langle \emptyset, \emptyset, \ldots, \emptyset \rangle$ be the least belief state of M. Note that it is not true, in general, that $\mathsf{S}(\emptyset) = \bot$, hence we cannot invoke the previous result.

However, $\bot \sqsubseteq \mathsf{S}(\emptyset)$, where $\sqsubseteq$ is pointwise set inclusion. By monotonicity of γ^2_M, it follows that $\gamma^{2n}_M(\bot) \sqsubseteq \gamma^{2n}_M(\mathsf{S}(\emptyset)) = \mathsf{S}\left(\gamma^{2n}_{\mathcal{KB}}(\emptyset)\right)$ by the previous result. Therefore, $\mathsf{lfp}\left(\gamma^2_M\right) \sqsubseteq \mathsf{S}\left(\mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right)\right)$.

From Corollary 2, we can write $\mathsf{lfp}\left(\gamma^2_M\right)$ as $\mathsf{S}(J)$ for some interpretation J, since γ_M is defined as the minimal equilibrium of a reduct of M. It follows that $\mathsf{S}(J) = \gamma^2_M(\mathsf{S}(J)) = \mathsf{S}\left(\gamma^2_{\mathcal{KB}}(J)\right)$, and hence J is a fixpoint of $\gamma^2_{\mathcal{KB}}$ (since S is injective). Therefore $\mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right) \sqsubseteq J$, and by monotonicity of S it follows that $\mathsf{S}\left(\mathsf{lfp}\left(\gamma^2_{\mathcal{KB}}\right)\right) \sqsubseteq \mathsf{S}(J) = \mathsf{lfp}\left(\gamma^2_M\right)$. □

4. Extensions and design options

In this section we extend our translation to dl-programs including the original $\uplus$ operator; and show how the mapping from dl-programs to MCSs can be used to motivate an encoding of default reasoning in this more general framework.

4.1. Non-monotonicity and the use of $\uplus$

As discussed in Section 2.2, the original definition of dl-programs [16] included a third operator $\uplus$ for building input contexts. The semantics for this operator is defined by extending the definition of $A_i(I)$ (page 6) with the clause

$$A_i(I) = \{\neg P_i(\bar{e}) \mid p_i(\bar{e}) \notin I\}$$

applicable when the input context χ includes $P_i \; op_i \; p_i$, where op_i is $\uplus$. Dl-atoms built using this operator are typically non-monotonic, as extending the interpretation of p reduces that of $P \uplus p$; this in turn implies that the definition of strong answer sets becomes more involved: the Gelfond–Lifschitz transformation also has to treat these non-monotonic atoms.

The same authors later observed [17] that $\uplus$ can be omitted from the language, under quite general assumptions (that the underlying description logic satisfies the unique name assumption), without restricting its expressiveness. Indeed, we can rewrite $P \uplus p$ as $P \cup \bar{p}$, where $\bar{p}$ is a fresh predicate symbol defined by means of the dl-rule

$$\bar{p}(X) \leftarrow \textbf{not} \;\; DL[Q \uplus p; Q](X)$$

(where Q is a fresh concept or role name), thereby "pushing" the non-monotonic component of the rule to the logic program side and making all dl-atoms monotonic.

As the authors observe in [17], this translation preserves the semantics of dl-programs, in the sense that applying this transformation does not change the set of models. For this reason, in that work they also treat only the fragment of the language we considered in this paper. In particular, they define well-founded semantics of dl-programs using only $\uplus$ and $\uplus$ (which they refer to as the "most relevant fragment" of the full language). The results informally stated in that article can be summarized in the following lemma.

Lemma 2. *Let $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ be a dl-program using all three operators $\uplus$, $\uplus$ and $\owedge$, and $\mathcal{KB}' = \langle \Sigma, \mathcal{P}' \rangle$ be the program using only $\uplus$ and $\uplus$ obtained by the transformation described above. Then I is a model of $\mathcal{KB}$ iff $\hat{I}$ is a model of $\mathcal{KB}'$, where $\hat{I}$ is obtained from I by defining the interpretation of $\bar{p}$ to be the complement of the interpretation of p, for every predicate p used in a non-monotonic subterm.*

Furthermore, we also prove the following relationship.

Lemma 3. *In the conditions of Lemma 2, if $\hat{I}$ is a strong answer set for $\mathcal{KB}'$, then I is a strong answer set for $\mathcal{KB}$.*

Proof. Let I be an interpretation and $\hat{I}$ be defined as above. Without loss of generality, assume that $\mathcal{P}$ is a ground dl-program. We first focus on the predicates p for which we add $\bar{p}$ in constructing $\mathcal{P}'$. In the reduct ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$, we include exactly

$$\bar{p}(\bar{t}) \leftarrow$$

for those $\bar{t}$ for which $p(\bar{t}) \notin I$, hence in every model of ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$ the interpretation of $\bar{p}$ must coincide with that defined in $\hat{I}$.

Furthermore, $I \models s\mathcal{P}^I_\Sigma$ iff $\hat{I} \models {s\mathcal{P}'}^{\hat{I}}_{\Sigma}$: since I and $\hat{I}$ agree on all dl-atoms, the rules that were kept in ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$ but have no counterpart in $s\mathcal{P}^I_\Sigma$ include literals in their body that are not satisfied by $\hat{I}$ (so those rules are always satisfied by I). As for the rules that were kept both in ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$ and $s\mathcal{P}^I_\Sigma$, the literals that were removed in the latter but not in the former are satisfied by $\hat{I}$, while satisfaction of the remaining literals coincides for I and $\hat{I}$.

Assume that I is not a strong answer set for $\mathcal{KB}$; we show that $\hat{I}$ is not a strong answer set for $\mathcal{KB}'$. If I is not a model of $s\mathcal{P}^I_\Sigma$, then we already know that $\hat{I}$ is not a model of ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$, so it cannot be a strong answer set. So assume that I is a model of $s\mathcal{P}^I_\Sigma$, but there is another model $J \subsetneq I$ of that program. Define J^I as the interpretation J extended to $\mathsf{HB}_{\mathcal{KB}'}$ by interpreting each symbol $\bar{p}$ as in $\hat{I}$. By construction $J^I \subsetneq \hat{I}$, and we show that J^I is also a model of $\mathsf{HB}_{\mathcal{KB}'}$. For the rules of the form $\neg p \leftarrow$ this is trivial. Now let r be a dl-rule in $\mathcal{P}$ that is kept in both $s\mathcal{P}^I_\Sigma$ and in ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$. Since J is a model of $s\mathcal{P}^I_\Sigma$, either J satisfies the head of r (and therefore so does J^I) or J does not satisfy one of the literals in the body of r that were kept in $s\mathcal{P}^I_\Sigma$; but such a literal cannot contain $\owedge$, therefore it appears unchanged in the body of the corresponding rule ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$ and is not satisfied by J^I. Finally we consider the case where r is removed from $s\mathcal{P}^I_\Sigma$ but not from ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$. This means that there is a (non-negated) dl-atom b in the body of r such that $I \not\models b$. But in this case also $\hat{I} \not\models b'$, where b' is the dl-atom obtained from b by the transformation defined above, and since $J^I \subsetneq \hat{I}$ and b' is a monotonic dl-atom it follows that $J^I \not\models b'$, whence J^I satisfies the counterpart to r in ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$. Therefore $J^I \models {s\mathcal{P}'}^{\hat{I}}_{\Sigma}$, and therefore $\hat{I}$ is not a strong answer set for ${s\mathcal{P}'}^{\hat{I}}_{\Sigma}$. □

The converse is however not true, as the following example shows.

Example 15. *Consider the following ground dl-program over an empty description logic knowledge base and a first-order signature containing only one constant symbol t.*

$$a(t) \leftarrow DL[P_1 \owedge b, P_2 \uplus c; \neg P_1 \cap P_2](t)$$
$$c(t) \leftarrow DL[P_1 \owedge d, P_2 \uplus a; \neg P_1 \cap P_2](t)$$

Then $I = \{a(t), c(t)\}$ is a strong answer set for this program (its reduct simply contains the two facts $a(t)$ and $c(t)$), but $\hat{I} = \{a(t), \bar{b}(t), c(t), \bar{d}(t)\}$ is not a strong answer set for the transformed program

$$a(t) \leftarrow DL[P_1 \uplus \bar{b}, P_2 \uplus c; \neg P_1 \cap P_2](t) \qquad \bar{b}(X) \leftarrow \mathbf{not}\ \ DL[B \uplus b; B](X)$$
$$c(t) \leftarrow DL[P_1 \uplus \bar{d}, P_2 \uplus a; \neg P_1 \cap P_2](t) \qquad \bar{d}(X) \leftarrow \mathbf{not}\ \ DL[D \uplus d; D](X)$$

as this program's reduct w.r.t. $\hat{I}$ is

$$a(t) \leftarrow DL[P_1 \uplus \bar{b}, P_2 \uplus c; \neg P_1 \cap P_2](t) \qquad \bar{b}(X) \leftarrow$$
$$c(t) \leftarrow DL[P_1 \uplus \bar{d}, P_2 \uplus a; \neg P_1 \cap P_2](t) \qquad \bar{d}(X) \leftarrow$$

which admits $\{\bar{b}(t), \bar{d}(t)\}$ as its minimal model.

We can use the same idea to extend the translation from dl-programs to MCSs to the full-fledged language also including the third operator $\barwedge$. This requires simply adding bridge rules $\neg P_j \leftarrow \mathbf{not}(0 : p_j)$ to the definition of $\mathsf{Ctx}(\Sigma)$ whenever op_i is $\barwedge$ (item 3 in Definition 7).

This construction essentially amounts to translating the dl-program first to a dl-program without $\barwedge$ and then translating the result to a multi-context system (albeit avoiding the addition of an explicit name for negated predicates). Therefore, Theorems 1 and 2 still hold. However, Theorem 3 does not hold, in view of Example 15), and we only have the following weaker property.

Proposition 5. *Let $\mathcal{KB} = \langle \Sigma, \mathcal{P} \rangle$ be a dl-program using all three operators $\uplus$, $\uplus$ and $\barwedge$, $\mathsf{M}(\mathcal{KB})$ be the corresponding MCS, and I be an interpretation over $\mathsf{HB}_{\mathcal{KB}}$. If $\mathsf{S}(\mathsf{I})$ is a grounded equilibrium of $\mathsf{M}(\mathcal{KB})$, then I is a strong answer set for $\mathcal{KB}$.*

Proof. Apply Theorem 3 to Lemma 3. □

Finally, it does not make sense to explore Theorem 4, since well-founded semantics for dl-programs are only defined for the fragment containing only $\uplus$ and $\uplus$ [17]. If we take the view that those authors implicitly defined these semantics for all dl-programs by means of this transformation, then of course Theorem 4 trivially holds.

4.2. Knowledge bases as contexts

As discussed in the introduction, having a translation from dl-programs to multi-context systems allows us in principle to take a systematic construction over dl-programs, translate it to the MCS framework, and examine whether it can be expressed as a general construction over MCSs. In this section, we undertake such a task to the encoding of default rules in dl-programs [18]. We show how it induces an encoding of default rules in MCSs, and prove that it correctly captures Reiter's extension semantics in the setting of ontologies.

In Reiter's original formulation [33], a default rule has the form

$$\frac{\alpha_1, \ldots, \alpha_k : \beta_1, \ldots, \beta_n}{\gamma}, \tag{1}$$

where α_i, β_j and γ are first-order formulas for all i, j, with intended semantics that if, for some instantiation θ of the free variables in the rule, all $\alpha_i\theta$ hold and it is consistent to assume that all $\beta_j\theta$ hold, then $\gamma\theta$ is inferred. Several semantics for default rules have been proposed [1], namely Reiter's original semantics [33] based on *extensions* – theories that are fixpoints w.r.t. the default rules. The presence of existential quantifiers in these formulas requires them to be skolemized before extensions are computed, which poses several problems namely in the setting of description logics [3]. Therefore, it is common to pose stronger syntactic restrictions on them; in particular, the results in [18] are first proven for literals only, although the authors then discuss how to relax this constraint to quantifier-free formulas where γ does not contain disjunctions.

In our setting, we are interested in modelling default rules by bridge rules. More precisely, following the idea in [18], we can encode rule (1) as the bridge rule $\gamma \leftarrow (i : \alpha_1), \ldots, (i : \alpha_k), \mathbf{not}(i : \neg\beta_1), \ldots, \mathbf{not}(i : \neg\beta_n)$ in br_i, where i is the identifier of the context $\mathsf{Ctx}(\Sigma)$. This suggests that $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_n$ should be dl-queries, while γ should be a formula that can occur in a knowledge base.

Definition 10. *A default rule over an ontology Σ is a rule of the form* (1)*, where α is a conjunction of dl-queries, $\beta_1, \ldots, \beta_n$ are dl-queries and γ is an assertion axiom.*

Recall that an assertion axiom is of the form $C(t)$, $\neg C(t)$, $R(t_1, t_2)$, $\neg R(t_1, t_2)$, $t_1 = t_2$ or $t_1 \neq t_2$, where C is a concept, R is a role, and t, t_1 and t_2 are constants or variables. This definition captures the intuition that default rules should only be used to infer additional instances to be added to the A-Box [3], as well as the restriction that γ should be a literal [18].

In order to define extensions, we again follow [3] and use Reiter's fixpoint characterization [33]. Given an ontology Σ, a set of default rules Γ and a set F of assertion axioms, let $\mathcal{E}(F, 0) = \Sigma$ and $\mathcal{E}(F, i+1)$ is obtained from $\mathcal{E}(F, i)$ by adding all γ such that: $\dfrac{\alpha_1 \wedge \ldots \wedge \alpha_k : \beta_1, \ldots, \beta_n}{\gamma} \in \Gamma$, $\mathcal{E}(F, i) \models \alpha_j$ for $1 \leq j \leq k$, and $\neg\beta_j \notin F$ for $1 \leq j \leq n$. Let $\mathcal{E}_F = \bigcup_{i=0}^{+\infty} \mathcal{E}(F, i)$.

Definition 11. *Let $\mathsf{Th}(E)$ denote the set of queries satisfied by an ontology E. An* extension *of Σ and Γ is a set $\mathsf{Th}(E)$ such that $\mathsf{Th}(E) = \mathsf{Th}(\mathcal{E}_E)$.*

Defining an extension as a set of queries (rather than a set of formulas) makes sense, as these are the only formulas that our framework allows us to "observe".

The result below makes the correspondence with the standard default semantics precise, again considering minimal equilibria.

Proposition 6. *Let Σ be an ontology and Γ be a set of default rules over Σ. Let M be the MCS with a single context $\mathsf{Ctx}(\Sigma)$ and bridge rules obtained from the rules in Γ. Then S is a minimal equilibrium of M iff S is an extension of Σ and Γ.*

Proof. Let S be an extension of Σ and Γ. Then $S = ACC_{\mathcal{L}}(S \cup \{H(r) \mid r \in br_i \text{ is applicable in } S\})$, hence S is an equilibrium of M. Conversely, if S is an equilibrium of M, then by induction $\mathcal{E}(S, i) \subseteq S$: for $i = 0$ this is trivial; for $i + 1$ assume that $\mathcal{E}(S, i) \subseteq S$ and note that $\mathcal{E}(S, i+1)$ is then derived from $\mathcal{E}(S, i)$ and the heads of rules that are applicable in $\mathcal{E}(S, i)$. Hence $\mathcal{E}_S \subseteq S$.

Then: if S is an equilibrium of M, then $\mathcal{E}_S \subseteq S$ is also an equilibrium of M that is simultaneously an extension of Σ and Γ; if S is a minimal equilibrium, then necessarily $\mathcal{E}_S = S$, yielding the thesis. Conversely, if S is an extension of Σ and Γ, then it is an equilibrium of M, and since extensions are minimal w.r.t. set inclusion it must be a minimal equilibrium. □

This result can be made a bit stronger; under some conditions, which often arise in practice, only minimal equilibria exist.

Corollary 3. *Let Σ, Γ and M be as in Proposition 6 and suppose that, for every extension E and rule $\dfrac{\alpha_1, \ldots, \alpha_k : \beta_1, \ldots, \beta_n}{\gamma} \in \Gamma$, $\alpha_i \in E$ iff α_i is a consequence of Σ. Then every equilibrium of M is an extension of Σ and Γ.*

Proof. Under the hypothesis, $\alpha_j \in \mathcal{E}(\Sigma, i)$ iff $\alpha_j \in E$, and the thesis follows. □

In particular, if the rules in Γ are *prerequisite free* [6] (i.e. $k = 0$), then every equilibrium of M corresponds to an extension of Σ and Γ, and conversely. This is interesting in practice, as it corresponds to many useful applications such as the modeling of closed-world reasoning by means of default rules [6].

Example 16. *Suppose that Σ is the ontology consisting of concepts C, D and E, the axiom $\mathsf{E} \equiv \mathsf{C} \sqcup \mathsf{D}$, and the single instance $\mathsf{E}(\mathsf{a})$. Then Σ's models must contain at least one of $\mathsf{C}(\mathsf{a})$ or $\mathsf{D}(\mathsf{a})$. Since none of these is guaranteed to hold in all models, $ACC_{\mathcal{L}}(\Sigma) = \emptyset$. Adding closed-world semantics to C and D, by means of the translated default rules $(1 : \neg\mathsf{C}(X)) \leftarrow \mathbf{not}(1 : \mathsf{C}(X))$ and $(1 : \neg\mathsf{D}(X)) \leftarrow \mathbf{not}(1 : \mathsf{D}(X))$, yields two possible equilibria, corresponding to the two extensions of the corresponding set of default rules: $\{\mathsf{C}(\mathsf{a}), \neg\mathsf{D}(\mathsf{a})\}$ and $\{\mathsf{D}(\mathsf{a}), \neg\mathsf{C}(\mathsf{a})\}$.*

In order to obtain *true* closed-world reasoning (in the sense that e.g. the MCS in Example 16 would be inconsistent, as Σ is inconsistent with the closed-world assumption) one could define ACC_i as a binary operator, separating the original belief state from the conclusions derived from the application of bridge rules and allowing them to be treated differently. We are currently studying the impact of this change in the theory of MCSs.

5. Conclusions

In this paper, we showed that an arbitrary dl-program can be faithfully translated into a multi-context system, despite the fundamental differences between these two paradigms for combining reasoning systems. This translation induces a precise correspondence between different types of semantics for dl-programs and MCSs. Thus, interpretations of the dl-program naturally correspond to belief states for the generated MCS; models correspond to equilibria; least models to minimal equilibria; strong answer sets to grounded equilibria; and well-founded semantics (for dl-programs) to well-founded semantics (for MCSs).

As a consequence of this construction, we are able to *compute* minimal equilibria and well-founded semantics for an MCS that has been generated from a dl-program. This is not possible in general, since the definition of minimal equilibrium is a declarative characterization that is not computational. Likewise, while there is no algorithmic procedure to check that an equilibrium for an MCS is grounded, this is possible for MCSs generated from dl-programs. This shows that the translation we presented is not only of theoretical interest, but can also be used to derive useful practical applications.

Our translation is initially restricted to the "most relevant fragment" of the language [17]: dl-programs where the view of the description logic knowledge base can only be updated by means of the operators $\uplus$ and $\dot{\cup}$. This is also the fragment for which well-founded semantics is defined. We show how the translation can be extended to the full-fledged language also including the operator $\curlywedge$, albeit by losing the precise correspondence between strong answer sets and grounded equilibria (which only holds in one direction). We apply standard techniques to encode $\curlywedge$ in the $\{\uplus, \dot{\cup}\}$-fragment of dl-programs, and we establish more precise properties of this encoding.

The precise correspondences between semantics crucially depend on our definition of context induced by a description logic knowledge base, which uses the set of known consequences of a knowledge base as semantics. To illustrate the relevance of this aspect, we study the encoding of default rules into MCSs (given by applying our translation to an encoding of default rules in dl-programs [18]). This example also shows how the mapping between dl-programs and MCSs can be used to translate systematic constructions from one of those frameworks to the other, widening the scope of their application: default rules in dl-programs can only span over one knowledge base, while default rules in MCSs can range over several contexts (not necessarily based on description logics).

Acknowledgements. This work was partially supported by a centre grant from FCT/MCTES/PIDDAC, Portugal (to BioISI, Centre Reference: UID/MULTI/04046/2013).

References

[1] G. Antoniou. A tutorial on default logics. *ACM Computing Surveys*, 31(3):337–359, 1999.

[2] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications. 2nd Edition.* Cambridge University Press, 2007.

[3] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[4] G. Brewka and T. Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *AAAI2007*, pages 385–390. AAAI Press, 2007.

[5] G. Brewka, T. Eiter, M. Fink, and A. Weinzierl. Managed multi-context systems. In T. Walsh, editor, *IJCAI*, pages 786–791. IJCAI/AAAI, 2011.

[6] G. Brewka, I. Niemel, and M. Truszczyński. Nonmonotonic reasoning. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 6, pages 239–284. Elsevier, 2008.

[7] G. Brewka, F. Roelofsen, and L. Serafini. Contextual default reasoning. In M.M. Veloso, editor, *IJCAI2007*, pages 268–273, 2007.

[8] J. Bruijn, T. Eiter, and H. Tompits. Embedding approaches to combining rules and ontologies into autoepistemic logic. In G. Brewka and J. Lang, editors, *KR2008*, pages 485–495. AAAI Press, 2008.

[9] L. Cruz-Filipe, G. Gaspar, and I. Nunes. Information flow within relational multi-context systems. In K. Janowicz, S. Schlobach, P. Lambrix, and E. Hyvönen, editors, *EKAW 2014*, volume 8876 of *LNAI*, pages 97–108. Springer, 2014. DOI 10.1007/978-3-319-13704-9_8.

[10] L. Cruz-Filipe, G. Gaspar, and I. Nunes. Design patterns for description-logic programs. In A. Fred, J.L.G. Dietz, K. Liu, and J. Filipe, editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 454 of *CCIS*, pages 199–214. Springer, 2015.

[11] L. Cruz-Filipe, R. Henriques, and I. Nunes. Description logics, rules and multi-context systems. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *LPAR-19*, volume 8312 of *LNCS*, pages 243–257. Springer, December 2013. DOI 10.1007/978-3-642-45221-5_18.

[12] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[13] M. Dao-Tran, T. Eiter, M. Fink, and T. Krennwallner. Dynamic distributed nonmonotonic multi-context systems. In G. Brewka, V. Marek, and M. Truszczynski, editors, *Nonmonotonic Reasoning, Essays Celebrating its 30th Anniversary*, volume 31 of *Studies in Logic*. College Publications, 2011.

[14] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.

[15] T. Eiter, M. Fink, G. Ianni, and P. Schüller. Towards a policy language for managing inconsistency in multi-context systems. In A. Mileo and M. Fink, editors, *Workshop on Logic-based Interpretation of Context: Modelling and Applications*, pages 23–35, 2011.

[16] T. Eiter, G. Ianni, T. Krenwallner, and A. Polleres. Rules and ontologies for the semantic web. In *Reasoning Web*, volume 5224 of *LNCS*, pages 1–53. Springer, 2008.

[17] T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the semantic Web. *ACM Transactions on Computational Logic*, 12(2), 2011. Article Nr 11.

[18] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12–13):1495–1539, 2008.

[19] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits. Reasoning with rules and ontologies. In P. Barahona, F. Bry, E. Franconi, N. Henze, and U. Sattler, editors, *Reasoning Web, Second International Summer School 2006, Lisbon, Portugal, September 4-8, 2006, Tutorial Lectures*, volume 4126 of *LNCS*, pages 93–127. Springer, September 2006.

[20] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Kaelbling and Saffiotti [30], pages 90–96.

[21] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. dlvhex: A prover for semantic-web reasoning under the answer-set semantics. In *Web Intelligence*, pages 1073–1074. IEEE Computer Society, 2006.

[22] M. Fink, L. Ghionna, and A. Weinzierl. Relational information exchange and aggregation in multi-context systems. In J.P. Delgrande and W. Faber, editors, *LPNMR*, volume 6645 of *LNCS*, pages 120–133. Springer, 2011.

[23] M. Fink and D. Pearce. A logical semantics for description logic programs. In T. Janhunen and I. Niemelä, editors, *JELIA 2010*, volume 6341 of *LNCS*, pages 156–168. Springer, 2010.

[24] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.

[25] R. Gonçalves, M. Knorr, and J. Leite. Evolving bridge rules in evolving multi-context systems. In N. Bulling, L.W.N. van der Torre, S. Villata, W. Jamroga, and W.W. Vasconcelos, editors, *CLIMA*, volume 8624 of *LNCS*, pages 52–69. Springer, 2014.

[26] R. Gonçalves, M. Knorr, and J. Leite. Evolving multi-context systems. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *PAIS*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 375–380. IOS Press, 2014.

[27] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNCS*, pages 701–706. Springer, 2001.

[28] S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with DL-programs over Datalog-rewritable description logics. In H. Coelho, R. Studer, and M. Wooldridge, editors, *ECAI2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 35–40. IOS Press, 2010.

[29] M. Homola, M. Knorr, J. Leite, and M. Slota. MKNF knowledge bases in multi-context systems. In M. Fisher, L. van der Torre, M. Dastani, and G. Governatori, editors, *CLIMA*, volume 7486 of *LNCS*, pages 146–162. Springer, 2012.

[30] L.P. Kaelbling and A. Saffiotti, editors. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30–August 5, 2005*. Professional Book Center, 2005.

[31] J. McCarthy. Notes on formalizing context. In R. Bajcsy, editor, *IJCAI1993*, pages 555–562. Morgan Kaufmann, 1993.

[32] B. Motik and R. Rosati. Reconciling description logics and rules. *Journal of the ACM*, 57, June 2010. Article Nr 30.

[33] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[34] F. Roelofsen and L. Serafini. Minimal and absent information in contexts. In Kaelbling and Saffiotti [30], pages 558–563.

[35] R. Rosati. DL+log: Tight integration of description logics and disjunctive Datalog. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *KR2006*, pages 67–78. AAAI Press, 2006.

[36] Y. Wang, J. You, L. Yuan, Y. Shen, and T. Eiter. Embedding description logic programs into default logic. *CoRR*, abs/1111.1486, 2011.