Here-and-There with Constraints for Description Logic

Pedro Cabalar[3][0000−0001−7440−0953], Stefania Costantini[2][0000−0002−5686−6124], and Alina Vozna[1,2][0009−0009−0179−6948]

[1] University of Pisa, Largo B. Pontecorvo, Pisa, 56127, Italy
[2] University of L'Aquila, Department of Information Engineering, Computer Science and Mathematics, Italy alina.vozna@phd.unipi.it, stefania.costantini@univaq.it
[3] University of A Coruña, Spain cabalar@udc.es

Abstract. In this short paper we introduce a theoretical framework for hybrid systems based on Answer Set Programming (ASP) modulo Description Logic (DL) theories. To this aim, we extend the formalism of Here-and-There with Constraints, mainly used before for integrating ASP with arithmetic constraints, to incorporate now theory atoms that may assert axioms or perform queries and reasoning on an external DL knowledge base. This paves the way for an alternative implementation of hybrid ASP systems interacting with large-scale existing ontologies that are treated as external theories.

Keywords: Answer Set Programming · Description Logic · Equilibrium Logic · ASP modulo theories.

1 Introduction

Answer Set Programming (ASP) [8,9,10] and *Description Logics* (DL) [1] constitute two well-known successful formalisms for practical Knowledge Representation and Reasoning. While ASP provides a highly expressive, non-monotonic reasoning framework well-suited for modeling complex combinatorial problems and default reasoning, DL forms the logical foundation of ontologies, enabling structured representation and reasoning about domain knowledge. Several approaches have considered the implementation of hybrid systems combining ASP and DL to leverage the benefits from both approaches. Some authors, for instance, have extended the ASP semantics to incorporate DL expressions, as in `OntoDLV` [11] or have used higher-order atoms in ASP to connect to DL theories as a particular case, as done in `DLVHEX` [6]. Other approaches have focused on logical formalisations, such as the one based on First Order Equilibrium Logic [2,12]. Another approach has consisted in defining parametrized logic programs, so other logics can be used as external theories [7].

In this paper, we consider a mixed approach, following a similar spirit to parametrized logic programs, but tackling for a logical formalisation of arbitrary formulas as in equilibrium logic. In particular, we introduce an instance of the logic of Here-and-There with Constraints [3], which was previously used for hybrid systems combining ASP

with arithmetic constraints, and extend it with new constraint atoms to refer to DL axioms. This approach follows the same spirit as SAT modulo theories, so the reasoning tasks for the external theories are treated as a blackbox from the ASP perspective.

Unlike existing hybrid ASP–DL approaches, our proposal does not introduce new DL-specific constructs into ASP nor does it tightly couple the ASP solver with a DL reasoner. Instead, Description Logic is treated as an external constraint theory within the HT_c framework, where DL axioms and queries are represented as constraint atoms with an externally defined semantics. This provides a uniform logical characterization of ASP modulo DL theories while remaining compatible with the general HT_c framework and its existing implementations. This paves the way for a future implementation of a DL-hybrid system that exploits the already existing technology in `clingo` for other hybrid systems [4] for arithmetics such as `clingo[DL]`, `clingcon` or `clingo[LP]`.

2 Background

In this section, we recall the basic definitions of the logic of *Here-and-There with constraints* (HT_c) mostly following [3]. We define the *signature* as a triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{X}$ is a set of *variables*, $\mathcal{D}$ is a set of values called the *domain* and $\mathcal{C}$ is a set of *constraint atoms*. A constraint atom represents a restriction on the relation among values of some variables, according to the semantics of some external theory. In the general case, constraints can be represented by arbitrary strings of symbols and do not need to have any structure. However, in most interesting cases, they usually have some structured form. As an example, constraint atom from *difference logic* has the general form "$x - y \leq d$" where x and y are integer variables and d is some integer constant. Although constraint atoms can be arbitrary strings, we assume that they may include, at least, occurrences of variables, like $x, y \in \mathcal{X}$ above, that act as placeholders. The set of all variables occurring in constraint atom c is denoted by $vars(c) \subseteq \mathcal{X}$. Let $\mathbf{u} \notin D$ be a distinguished symbol representing an undefined value, and let $D_{\mathbf{u}} = D \cup \{\mathbf{u}\}$ denote the extended domain. Notice that constraint atoms c with $vars(c) = \emptyset$ just amount to a truth constant $\bot$ or $\top$, and can be disregarded. We write $c[s/s']$ to represent the syntactic replacement of subexpression s by s' in constraint atom c. We also assume that $c[x/d] \in \mathcal{C}$ for any constraint atom $c[x] \in \mathcal{C}$, variable $x \in \mathcal{X}$ and $d \in \mathcal{D}_{\mathbf{u}}$. That is, the constraint that results from replacing any variable x by any element d of the extended domain is also a syntactically valid constraint atoms. In the previous example, if "$x - y \leq 5$" is a constraint then "$2 - y \leq 5$" and "$\mathbf{u} - 2 \leq 5$" must also be valid constraints, to put a pair of examples.

A *valuation* v over $\mathcal{X}, \mathcal{D}$ is a partial function from variables in $\mathcal{X}$ to domain elements in $\mathcal{D}$. The set of valuations over $\mathcal{X}$ and $\mathcal{D}$ is denoted by $\mathcal{V}^{\mathcal{X},\mathcal{D}}$, or just $\mathcal{V}$ when there is no ambiguity. Since v is a partial function, it may leave some variables $x \in \mathcal{X}$ undefined, something we will represent as $v(x) = \mathbf{u}$ where $\mathbf{u} \notin \mathcal{D}$ is a new symbol not included in the domain. Using that extended domain, a valuation can be seen as a total function of the form $v : \mathcal{X} \to \mathcal{D}_{\mathbf{u}}$. Alternatively, we may also represent v as the set of pairs $\{x \mapsto v(x) \mid x \in \mathcal{X}, v(x) \in \mathcal{D}\}$ where each x is assigned at most one domain value $v(x) \in \mathcal{D}$ (remember $\mathbf{u} \notin \mathcal{D}$). Under this set of pairs representation, we can then

define the subset relation $v \subseteq v'$ between two interpretations v, v' in the standard way. It is not hard to see that $v \subseteq v'$ amounts to the condition, for all $x \in \mathcal{X}$: either $v(x) = \mathbf{u}$ or $v'(x) = v(x)$.

The *domain* of a valuation v is the set of variables that have a defined value $dom(v) \stackrel{\text{def}}{=} \{x \in \mathcal{X} \mid v(x) \neq \mathbf{u}\}$. Given a valuation $v : \mathcal{X} \to \mathcal{D}_\mathbf{u}$ we overload its use for the extended definition $v : \mathcal{X} \cup \mathcal{D}_\mathbf{u} \to \mathcal{D}_\mathbf{u}$ fixing $v(d) = d$ for any $d \in \mathcal{D}$ and $v(\mathbf{u}) = \mathbf{u}$.

The semantics of a constraint atom $c \in \mathcal{C}$ is defined by its *denotation* $[\![c]\!] \subseteq \mathcal{V}$ that assigns to c some set of valuations following the conditions, for all $x \in \mathcal{X}$, $v, v' \in \mathcal{V}$:

1. $v \in [\![c]\!]$ and $v \subseteq v'$ imply $v' \in [\![c]\!]$,
2. $v \in [\![c]\!]$ implies $v \in [\![c[x/v(x)]]\!]$,
3. if $v(x) = v'(x)$ for all $x \in vars(c)$ then $v \in [\![c]\!]$ iff $v' \in [\![c]\!]$.

The first condition states that for any valuation v in $[\![c]\!]$ that leaves some variables undefined we must also have all possible valuations v' in $[\![c]\!]$ with all the possible ways to complete the undefined values. For instance, if $\mathcal{X} = \{x, y\}$, $\mathcal{D}$ and $v = \{x \mapsto 3\} \in [\![c]\!]$ then $v' \in [\![c]\!]$ for all $v' = \{x \mapsto 3, y \mapsto d\}$ for $d \in \mathcal{D}$. The second condition guarantees that variables act as placeholders: for instance, if $v(x) = 5$ and $v \in [\![c]\!]$ then v must also belong to $[\![c[x/5]]\!]$. Condition 3 asserts that only the variables in $vars(c)$ are relevant for the denotation of c, while the rest of variables may freely vary.

A *formula* φ over signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is defined by the grammar:

$$\varphi ::= \bot \mid c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \quad \text{where } c \in \mathcal{C}.$$

We define the derived operators $\top$ as $\bot \to \bot$ and $\neg\varphi$ as $\varphi \to \bot$. Also, we define $vars(\varphi) \stackrel{\text{def}}{=} \bigcup\{vars(c) \mid c \text{ occurs in } \varphi\}$. An HT_c*-theory* is a set of formulas.

An *interpretation* over $\mathcal{X}, \mathcal{D}$ is a pair $\langle h, t \rangle$ of valuations over $\mathcal{X}, \mathcal{D}$ satisfying $h \subseteq t$. We say that the interpretation is *total* when $h = t$.

Definition 1 (Satisfaction of formulas). *An interpretation* $\langle h, t \rangle$ satisfies *a formula* φ *with respect to some fixed denotation* $[\![\cdot]\!]$*, written* $\langle h, t \rangle \models \varphi$*, if the following recursive conditions hold:*

1. $\langle h, t \rangle \not\models \bot$
2. $\langle h, t \rangle \models c$ *if* $h \in [\![c]\!]$
3. $\langle h, t \rangle \models \varphi \wedge \psi$ *if* $\langle h, t \rangle \models \varphi$ *and* $\langle h, t \rangle \models \psi$
4. $\langle h, t \rangle \models \varphi \vee \psi$ *if* $\langle h, t \rangle \models \varphi$ *or* $\langle h, t \rangle \models \psi$
5. $\langle h, t \rangle \models \varphi \to \psi$ *if* $\langle w, t \rangle \not\models \varphi$ *or* $\langle w, t \rangle \models \psi$ *for* $w \in \{h, t\}$

An interpretation $\langle h, t \rangle$ is a *model* of a theory Γ, written $\langle h, t \rangle \models \Gamma$, when $\langle h, t \rangle \models \varphi$ for every $\varphi \in \Gamma$. Two theories of formulas Γ, Γ' are said to be equivalent $\Gamma \equiv \Gamma'$ when they have the same models. The following proposition provides some basic, useful properties of HT_c.

Proposition 1 (Proposition 3 in [5]). *For any formula* φ*, we have*

1. $\langle h, t \rangle \models \varphi$ *implies* $\langle t, t \rangle \models \varphi$*,*
2. $\langle h, t \rangle \models \neg\varphi$ *iff* $\langle t, t \rangle \not\models \varphi$*, and*
3. *any tautology in* HT *is also a tautology in* HT_c*.*

Although, in general, HT_c makes no assumptions about the syntactic form of constraint atoms, [4] introduced a specific case of constraint atom that is useful for proving formal results. Given some domain subset $\mathcal{D}' \subseteq \mathcal{D}$, we define an associated constraint atom $x : \mathcal{D}'$ with denotation:

$$[\![x : \mathcal{D}']\!] \stackrel{\text{def}}{=} \{v \in \mathcal{V}^{\mathcal{X},\mathcal{D}} \mid v(x) \in \mathcal{D}'\}$$

and $vars(x : \mathcal{D}') = \{x\}$. The formula $def(x)$ stands for $x : \mathcal{D}$ and so, it holds when $v(x) \neq \mathbf{u}$.

Nonmonotonicity is achieved by selecting a kind of minimal HT_c models, we call *equilibrium models* and define as follows.

Definition 2 (Equilibrium/Stable model). *A (total) interpretation $\langle t, t\rangle$ is an* equilibrium model *of a theory Γ, if $\langle t, t\rangle \models \Gamma$ and there is no $h \subset t$ such that $\langle h, t\rangle \models \Gamma$.*

If $\langle t, t\rangle$ is an equilibrium model of Γ, then we say that t is a stable model *of Γ.*

As explained in [5], we can capture the original logic of Here-and-There in HT_c by using a signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, where $\mathcal{X} = \mathcal{A}$ are the logical propositions, the domain $\mathcal{D} = \{\mathbf{t}\}$ contains a unique value (meaning "true") and the set of constraint atoms $\mathcal{C} = \mathcal{A}$ coincides with the set of propositions. In [4], a notation distinction was introduced to clarify that each logical proposition $\mathtt{a}$ is both a constraint atom $\mathtt{a} \in \mathcal{C}$ and a homonymous variable (using a different font) $a \in \mathcal{X}$ so that $vars(\mathtt{a}) = \{a\}$. The denotation of a regular atom is defined as:

$$[\![\mathtt{a}]\!] \stackrel{\text{def}}{=} [\![a : \{\mathbf{t}\}]\!] = \{v \in \mathcal{V}^{\mathcal{X},\mathcal{D}} \mid v(a) = \mathbf{t}\}$$

Under this representation, a regular atom can only be "true" or undefined (there is no need for an explicit value for "false"). A valuation for propositional atoms can be simply represented as a set of atoms $X = \{\mathtt{a} \mid v(a) = \mathbf{t}\}$. By doing so, it is easy to see that the definition of equilibrium and stable models in Definition 2 collapses to their standard definition for propositional theories (and so, for logic programs in ASP too).

In general, when we deal with hybrid systems, we will be interested in combining regular atoms with other types of constraint atoms. For this reason, we may assume that we deal with signatures containing at least the element $\mathbf{t} \in \mathcal{D}$, and some subset of variables $\mathcal{A} \subseteq \mathcal{X}$ including the regular atoms and satisfying the following axiom:

$$def(a) \rightarrow \mathtt{a} \tag{1}$$

for every $a \in \mathcal{A}$. This axiom states that if a has some value, it must be value $\mathbf{t}$, avoiding in this way that regular atoms may take other values from $\mathcal{D} \setminus \{\mathbf{t}\}$.

3 Description Logic HT_c

We proceed now to introduce an instance of HT_c where constraint atoms represent Description Logic axioms. In this setting, all previous definitions for HT_c are still maintained (including logical formulas and propositional atoms). We call HT_{dl} this HT_c subsystem. The HT_{dl} signature is a 4-tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \Omega\rangle$ where $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$ is a regular HT_c signature and the fourth component Ω is a (generally infinite) set of elements we call the *universe* from which multiple domains will be formed. Additionally, an HT_{dl} signature must satisfy the following conditions:

1. The set of variables $\mathcal{X} = \mathcal{A} \cup \mathcal{X}_C \cup \mathcal{X}_R \cup \mathcal{X}_O$ is partitioned into four disjoint sets: $\mathcal{A}$, the set of regular atoms; $\mathcal{X}_C$, the set of *concept names*; $\mathcal{X}_R$, the set of *role names*; and $\mathcal{X}_O$, the set of *object names*.
2. The domain $\mathcal{D} = \{\mathbf{t}\} \cup \mathcal{D}_C \cup \mathcal{D}_R$ consists again of four disjoint subsets: $\mathbf{t}$ is the element "true" for regular atoms; $\mathcal{D}_C = 2^{\Omega}$ are the possible "values" for concept variables in $\mathcal{X}_C$, being each value some set of elements; and $\mathcal{D}_R = 2^{\Omega \times \Omega}$ are the possible values for role variables in $\mathcal{X}_R$, where each value will be a set of pairs of elements from the universe. The domain for objects $\mathcal{D}_O$ is not needed as it coincides with Ω.
3. A constraint atom $c \in \mathcal{C}$ is recursively defined following the grammar:

$$c ::= o : E \mid (o_1, o_2) : R \mid E_1 \sqsubseteq E_2 \mid C \equiv E$$

   where $C \in \mathcal{X}_C$, $R \in \mathcal{X}_R$ denote any concept or role variable, respectively, $o, o_1, o_2 \in \mathcal{X}_O$ are constant names and E, E_1, E_2 are *concept expressions* following, in their turn, the grammar:

$$E ::= C \mid E_1 \sqcap E_2 \mid \forall R.E \mid \exists R.E \mid {\sim}E \mid \bot$$

The connectives allowed in concept expressions above correspond to description logic $\mathcal{ALC}$, but other operators covering more expressive description logics can be easily introduced. We use the symbol $\sim$ for negation in DL, to distinguish it from $\neg$ we use for HT (or default) negation.

A valuation v in HT$_{dl}$ is associated to a particular subset $\Delta_v \subseteq \Omega$ of the universe we call the *concept domain*. We will sometimes write Δ instead of Δ_v when clear from the context. An HT$_{dl}$-valuation v must satisfy, for all $x \in \mathcal{X}$ such that $v(x) \neq \mathbf{u}$:

- $v(x) = \mathbf{t}$ if $x \in \mathcal{A}$
- $v(x) \subseteq \Delta$ if $x \in \mathcal{X}_C$
- $v(x) \subseteq \Delta \times \Delta$ if $x \in \mathcal{X}_R$
- $v(x) \in \Delta$ if $x \in \mathcal{X}_O$

This means that defined regular atoms can only be assigned value $\mathbf{t}$, defined concept variables can only be assigned sets of elements from Δ and defined role variables can only be assigned sets of pairs of elements from Δ. Additionally, defined object variables can only be assigned an element value from Δ.

We extend the application of a valuation v not only to variables but also to any concept expression E (different from concept names) following the recursive conditions:

$$\begin{aligned}
v(\bot) &\stackrel{\text{def}}{=} \emptyset \\
v(E_1 \sqcap E_2) &\stackrel{\text{def}}{=} v(E_1) \cap v(E_2) \\
v(\forall R.E) &\stackrel{\text{def}}{=} \{d \in \Delta \mid \text{for all } d' \text{ s.t. } (d, d') \in v(R) : d' \in v(E)\} \\
v(\exists R.E) &\stackrel{\text{def}}{=} \{d \in \Delta \mid \text{there exists } d' \text{ s.t. } (d, d') \in v(R), d' \in v(E)\} \\
v({\sim}E) &\stackrel{\text{def}}{=} \Delta \setminus v(E)
\end{aligned}$$

Definition 3. *The denotation of a* HT_{dl}*-constraint atom* c*,* $[\![c]\!]$*, is recursively defined as follows:*

$$[\![o : E]\!] \stackrel{\text{def}}{=} \{v \mid v(o) \in v(E)\}$$
$$[\![(o_1, o_2) : R]\!] \stackrel{\text{def}}{=} \{v \mid (v(o_1), v(o_2)) \in v(R)\}$$
$$[\![E_1 \sqsubseteq E_2]\!] \stackrel{\text{def}}{=} \{v \mid v(E_1) \subseteq v(E_2)\}$$
$$[\![E_1 \equiv E_2]\!] \stackrel{\text{def}}{=} \{v \mid v(E_1) = v(E_2)\}$$

Although, in the general case in HT_{dl}, we may construct arbitrary propositional theories (formed with regular atoms and DL constraint atoms), we will be mostly interested in logic programs, that is, implications $\varphi \leftarrow \psi$ where φ is a disjunction of literals and ψ a conjunction of literals. Informally speaking, the effect produced by HT_{dl}-logic programs is that we may specify which set of axioms we wish to include in our DL theory by deriving them as constraint atoms in a logic program. Consider the following example:

Example 1. The default situation among the 192 countries in the UN is that same-sex marriage is not allowed. On the other hand, 40 countries have approved laws to allow same-sex marriages. Let $\mathcal{X}_\mathcal{C} = \{Man, Woman, Person\}$ and $\mathcal{X}_\mathcal{R} = \{married\}$, $\mathcal{X}_O = \{Mary\}$. Consider the following logic program:

$$Woman \sqcap \exists married.Woman \equiv \bot \leftarrow \neg law \tag{2}$$
$$Man \sqcap \exists married.Man \equiv \bot \leftarrow \neg law \tag{3}$$
$$\exists married.\top \sqsubseteq Person \tag{4}$$
$$\forall married.\ Person \equiv \top \tag{5}$$
$$Person \equiv Man \sqcup Woman \tag{6}$$
$$Mary : Woman \tag{7}$$
$$Mary : \exists married.\top \tag{8}$$

Rules (2) and (3) forbid same-sex marriages if we cannot prove that there exists a law that permits them. Facts (4) and (5) state that we can only marry persons, that are defined as men or women by fact 6. Finally, (7) and (8) assert that $Mary$ is a married woman. From (2)-(8) we obtain a set of equilibrium models (they vary depending on the domain Δ) that satisfy the same set of constraint atoms and from which we can conclude $Mary : \exists married.Man$, that is, $Mary$ must have a husband. On the other hand, if we add the fact law to the same logic program, the DL constraints in the heads of (2) and (3) are no longer applicable, so we cannot derive $Mary : \exists married.Man$ any more. The effect of adding law to the program is the same as if we consider the standard DL theory formed by the facts (4)-(8).

Example 2. Consider an extended example where we do not know the gender of some new element $Chris \in \mathcal{X}_O$. Yet, we may derive that some atom $known$ holds if we know that $Chris$ is a man or we know that $Chris$ is a woman. We can represent the

situation with the logic program:

$$Chris : Woman \vee Chris : Man \tag{9}$$

$$known \leftarrow Chris : Woman \tag{10}$$

$$known \leftarrow Chris : Man \tag{11}$$

From this program, we get two equilibrium models (modulo different domains Δ): one in which we derive $Chris : Woman$ and $known$, and the other one where we obtain $Chris : Man$ and $known$. This result corresponds to considering two different DL theories, depending on whether we include the DL axioms $Chris : Woman$ or $Chris : Man$, respectively. Note the difference with respect to replacing (9) with:

$$Chris : Woman \sqcup Man \tag{12}$$

In this case, we cannot derive fact $known$ any more. This is because once we state (12) instead of (9), we can no more derive the body $Chris : Man$ of rule (10) or the body $Chris : Woman$ of rule (11). As a result, atom $known$ cannot be derived either.

HT_{dl} allows some limited kind of non-monotonic reasoning for DL expressions.

Example 3. Consider the classical example stating that birds that normally fly:

$$AbBird \sqsubseteq Bird \tag{13}$$

$$AbBird \sqcap Fly \equiv \bot \tag{14}$$

$$Bird \sqcap {\sim}AbBird \sqsubseteq Fly \tag{15}$$

$$Tweety : {\sim}AbBird \leftarrow Tweety : Bird \wedge \neg(Tweety : AbBird) \tag{16}$$

$$Tweety : Bird \tag{17}$$

Rules (13) and (14) state that abnormal birds are birds and that they do not fly. Rule (15) says that a normal bird can fly. Rule (16) expresses that $Tweety$ is a normal bird by default, whereas (17) tells us that $Tweety$ is a bird. From this program, we can conclude that $Tweety : {\sim}AbBird$ and so $Tweety : Fly$. However, if we add the rules:

$$Penguin \sqsubseteq AbBird$$

$$Tweety : Penguin$$

we retract those facts and eventually conclude $Tweety : {\sim}Fly$.

We may note that, in the previous example, we should have used the more general rule:

$$X : {\sim}AbBird \leftarrow X : Bird \wedge \neg(X : AbBird)$$

instead of (16). However, this rule is not propositional, as it makes use of a (logic program) variable X standing for any element in the ASP (Herbrand) domain. Note that ASP variables are quite different from constraint variables: the former can be seen as an abbreviation for all possible instantiations with domain elements in the ASP program, whereas constraint variables are entities completely external to the ASP program and interpreted through denotations, as explained before. ASP variables in HT_{dl}-logic programs introduce new issues such as safety or the difference between the ASP domain, which is usually finite and follows the unique names assumption, versus the DL domain, which is infinite and does not make any assumption on names. We leave those problems for future study.

4 Conclusions

We have introduced a logic for hybrid systems based on Answer Set Programming (ASP) modulo Description Logic (DL) theories. To do so, we extended the formalism of Here-and-There with Constraints (which was previously used for integrating ASP with arithmetic constraints) to incorporate theory atoms capable of asserting axioms and performing queries and reasoning over an external DL knowledge base. Our future work will be focused on comparing with other existing approaches, extending the current formalism to theories/programs with variables, and eventually, on the implementation of a hybrid ASP system that allows interacting with large-scale existing ontologies.

Acknowledgements . We want to thank the anonymous reviewers for their useful suggestions to improve the paper. This work was supported by grant PID2023-148531NB-I00 funded by Spanish Ministry MCIU/AEI/10.13039/501100011033

References

1. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017), http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-description-logic?format=PB#17zVGeWD2TZUeu6s.97
2. de Bruijn, J., Pearce, D., Polleres, A., Valverde, A.: A semantical framework for hybrid knowledge bases. Knowledge Information Systems **25**(1), 81–104 (2010)
3. Cabalar, P., Fandinno, J., Schaub, T., Wanko, P.: An ASP semantics for constraints involving conditional aggregates. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain. pp. 664–671. Frontiers in Artificial Intelligence and Applications, IOS Press (2020)
4. Cabalar, P., Fandinno, J., Schaub, T., Wanko, P.: Strong equivalence in answer set programming with constraints. CoRR **abs/2502.04302** (2025). https://doi.org/10.48550/ARXIV.2502.04302
5. Cabalar, P., Kaminski, R., Ostrowski, M., Schaub, T.: An ASP semantics for default reasoning with constraints. In: Kambhampati, S. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. pp. 1015–1021. IJCAI/AAAI Press (2016)
6. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: dlvhex: A prover for semantic-web reasoning under the answer-set semantics. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. p. 1073–1074. WI '06, IEEE Computer Society, USA (2006)
7. Gonçalves, R.: Combining rules and ontologies via parametrized logic programs. In: Papini, O., Benferhat, S., Garcia, L., Mugnier, M., Fermé, E.L., Meyer, T., Wassermann, R., Hahmann, T., Baclawski, K., Krisnadhi, A., Klinov, P., Borgo, S., Kutz, O., Porello, D. (eds.) Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25-27, 2015. CEUR Workshop Proceedings, CEUR-WS.org (2015)
8. Lifschitz, V.: Answer set planning. In: Proc. ICLP-1999. pp. 23–37. MIT Press (1999)

9. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm, pp. 169–181. Springer-Verlag (1999)
10. Niemelä, I.: Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. Ann. Math. Artif. Intell. **25**(3-4), 241–273 (1999)
11. Ricca, F., Gallucci, L., Schindlauer, R., Dell'Armi, T., Grasso, G., Leone, N.: OntoDLV: An ASP-based system for enterprise ontologies. Journal of Logic and Computation **19**(4), 643–670 (2009)
12. Stefano, F.D., Simkus, M.: Equilibrium description logics: Results on complexity and relations to circumscription. In: Marquis, P., Ortiz, M., Pagnucco, M. (eds.) Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024 (2024)