Functional Stable Model Semantics, Answer Set Programming Modulo Theories, and Action Languages

Joohyung Lee

Automated Reasoning Group
Arizona State University, USA

NRAC 2013 Invited Talk

Abstract

"Answer Set Programming Modulo Theories (ASPMT)" is a recently proposed framework which tightly integrates answer set programming (ASP) and satisfiability modulo theories (SMT). Its mathematical foundation is the functional stable model semantics, an enhancement of the traditional stable model semantics to allow defaults involving functions as well as predicates. This talk will discuss how ASPMT can provide a way to overcome limitations of the propositional setting of ASP, how action language C+ can be reformulated in terms of ASPMT, and how it can be implemented based on the reformulation.

Introduction

Answer Set Programming (ASP)

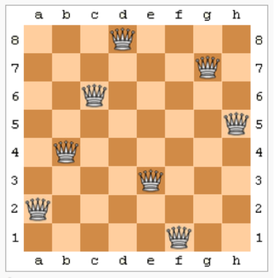
- Declarative programming paradigm. Suitable for knowledge intensive and combinatorial search problems.
- Theoretical basis: answer set semantics (Gelfond & Lifschitz, 1988).
- Expressive representation language: defaults, recursive definitions, aggregates, preferences, etc.
- ASP solvers:
  - SMODELS (Helsinki University of Technology, 1996)
  - DLV (Vienna University of Technology, 1997)
  - CMODELS (University of Texas at Austin, 2002)
  - PBMODELS (University of Kentucky, 2005)
  - CLASP (University of Potsdam, 2006) – winning first places at ASP'07/09/11/12, PB'09/11/12, and SAT'09/11/12
  - DLV-HEX computing HEX programs.
  - oClingo for reactive answer set programming.

Declarative Problem Solving using ASP

The basic idea is

- to represent the given problem by a set of rules,
- to find answer sets for the program using an ASP solver, and
- to extract the solutions from the answer sets.

N-Queens Puzzle in the Language of CLINGO




```
number(1..n).
#domain number(I).
#domain number(I1).
#domain number(J).
#domain number(J1).

% Each column has exactly one queen
1{q(K,J) : number(K)}1.

% Two queens cannot stay on the same row
:- q(I,J), q(I,J1), J<J1.

% Two queens cannot stay on the same diagonal
:- q(I,J), q(I1,J1), J<J1, #abs(I1-I)==J1-J.
```

Finding All Solutions for the 8-Queens Problem

With the command line

```
% clingo queens -c n=8 0
```

CLINGO computes and shows all 92 valid queen arrangements. For instance, the last one is

```
Answer: 92
q(5,8) q(7,7) q(2,6) q(6,5) q(3,4) q(1,3) q(8,2) q(4,1)
SATISFIABLE

Models       : 92
Time         : 0.010
  Prepare    : 0.000
  Prepro.    : 0.000
  Solving    : 0.010
```

Applications of ASP in AI

- planning ([Lif02], [DEF+03], [SPS09], [TSGM11], [GKS12])
- theory update/revision ([IS95], [FGP07], [OC07], [EW08], [ZCRO10], [Del10])
- preferences ([SW01], [Bre07], [BNT08a])
- diagnosis ([EFLP99], [BG03], [EBDT+09])
- learning ([Sak01], [Sak05], [SI09], [CSIR11])
- robotics ([CHO+09], [EHP+11], [AEEP11], [EHPU12], [APE12])
- description logics and semantic web ([EGRH06], [CEO09], [Sim09], [PHE10], [SW11], [EKSX12])
- data integration and question answering ([AFL10], [LGI+05])
- multi-agent systems ([VCP+05], [SPS09], [SS09], [BGSP10], [Sak11], [PSBG12])
- multi-context systems ([EBDT+09], [BEF11], [EFS11], [BEFW11], [DFS12])
- natural language processing/understanding ([BDS08], [BGG12], [LS12])
- argumentation ([EGW08], [WCG09], [EGW10], [Gag10])

Applications of ASP in Other Areas

- product configuration ([SN98], [TSNS03]) used by Variantum Oy
- Linux package configuration ([Syr00], [GKS11])
- wire routing ([ELW00], [ET01])
- combinatorial auctions ([BU01])
- game theory ([VV02], [VV04])
- decision support systems ([NBG+01]): used by United Space Alliance
- logic puzzles ([FMT02], [BD12])
- bioinformatics ([BCD+08], [EY09], [EEB10], [EEEO11])
- phylogenetics ([ELR06], [BEE+07], [Erd09], [EEEF09], [CEE11], [Erd11])
- haplotype inference ([EET09], [TE08])
- systems biology ([TB04], [GGI+10], [ST09], [TAL+10], [GSTV11])
- automatic music composition ([BBVF09],[BBVF11])
- assisted living ([MMB08], [MMB09], [MSMB11])
- team building ([RGA+12]): used by Gioia Tauro seaport
- software engineering ([EIO+11])
- bounded model checking ([HN03], [TT07])
- verification of cryptographic protocols ([DGH09])
- e-tourism ([RDG+10])

What Led to the Success of ASP?

- A simple, mathematically elegant semantics, based on the concept of a stable model
- Intelligent grounding—the process that replaces first-order variables with corresponding ground instances
- Efficient search methods that originated from propositional satisfiability solvers (SAT solvers).

Various Extensions

Starting from the Prolog syntax, the language of ASP has evolved:

- Strong negation [GLR91]
- Choice rules [SNS02]
- Aggregates [SNS02, FLP04, Fer05, PDB07, LM09, FL10], ...
- Preferences [BNT08b]
- Integration with CSP [Bal09, GOS09]
- Integration with SMT [JLN11]
- Integration with Description Logics [EIL+08, LP11]
- Probabilistic answer sets [BGR09]
- ...

Language Extension: Constraint Answer Set Programs (CASP)

Grounding is often the bottleneck. Solving is not applied until grounding is finished.

To alleviate the grounding bottleneck, integration of ASP with CSP/SMT solvers has been considered.

- Clingcon [GOS09]: Clasp + CSP solver Gecode

$$1 \leq amt(T) \leq 3 \leftarrow pour(T)$$
$$amt(T) = 0 \leftarrow not\ pour(T)$$
$$vol(T+1) = vol(T) + amt(T)$$

- EZCSP [Bal11]: Gringo + constraint solver SICStus Prolog or BProlog
- Dingo [JLN11]: Gringo + SMT solver Barcelogic

Fundamental Limitation Due to Lack of General Functions

ASP lacks general functions.

- Functional fluents in ASP are represented by predicates:

  $$WaterLevel(t+1, tank, l) \leftarrow WaterLevel(t, tank, l),\ not\ \ {\sim}WaterLevel(t+1, tank, l).$$

  Grounding generates a large number of instances as the domain gets larger.

- Using functions (e.g., $WaterLevel(t, tank) = l$) instead does not work because

  - Answer sets are Herbrand models: $WaterLevel(t+1, tank) = WaterLevel(t, tank)$ is always false.
  - Nonmonotonicity of ASP has to do with minimizing the predicates but has nothing to do with functions.

Fundamental Limitation Due to Lack of General Functions

- Even the constraint answer set sovers don't help. In CLINGCON this rule does not affect stable models.

$$\begin{aligned} &\mathit{WaterLevel}(t+1, \mathit{tank}) =^{\$} l \leftarrow \\ &\qquad \mathit{WaterLevel}(t, \mathit{tank}) =^{\$} l,\ \mathit{not}\ \mathit{WaterLevel}(t+1, \mathit{tank}) \neq^{\$} l. \end{aligned}$$

The lack of general functions in ASP is not only a disadvantage in comparison with other KR formalisms, but also a hurdle to cross over in integrating ASP with other declarative paradigms where functions are primitive constructs.

From Wikipedia Article on SMT

By comparison, answer set programming is also based on predicates *(more precisely, on atomic sentences created from atomic formula). Unlike SMT, answer-set programs do not have quantifiers, and cannot easily express constraints such as linear arithmetic or difference logic–ASP is at best suitable for boolean problems that reduce to the free theory of uninterpreted functions.*

ASP vs. SMT

- ASP is a successful nonmonotonic declarative programming paradigm, but is limited in handling first-order reasoning involving functions due to its propositional setting.
- SMT is a successful approach to solving some specialized first-order reasoning, but is limited in handling expressive nonmonotonic reasoning.

Answer Set Programming Modulo Theories (ASPMT)

- ASPMT tightly integrates ASP and SMT:

Monotonic	Nonmonotonic
FOL	Functional Stable Model Semantics [BL12]
SMT	ASP Modulo Theories [BL13]
SAT	Traditional ASP

- ASP = SAT + Loop formulas
- The syntax of ASPMT is the same as that of SMT. The semantics is defined as a special case of FSM [BL12].
- $$WaterLevel(t+1, tank) = l \leftarrow$$
  $$WaterLevel(t, tank) = l,\ \ not\ \ WaterLevel(t+1, tank) \neq l$$
  works under ASPMT.

My View

- The stable model semantics was successfully extended to the first-order level.
- Its computation can be carried out by compilation to declarative solvers, such as ASP solvers, CSP solvers, SMT solvers, ontology reasoners, or their combinations.
- Like SMT, we need to restrict to certain classes of first-order reasoning. ASP modulo theories will enjoy the expressiveness of the ASP modeling language while leveraging efficient constraint / theory solving methods available in SMT and other related computing paradigms.

This Talk is About

- First-order stable model semantics (FOSM).
- Extending FOSM to allow intensional functions
- ASPMT
- Reformulating $\mathcal{C}+$ in ASPMT

First-Order Stable Model Semantics

First-Order Stable Model Semantics [FLL07, FLL11]

Generalizes Gelfond and Lifschitz's 1988 definition of a stable model to first order sentences.

- Does not refer to grounding; not restricted to Herbrand models.
- Does not refer to reduct.
- Defined by a translation into second-order classical logic.

The stable models of F are defined as the models of F (in the sense of classical logic) that satisfy the "stability condition."

First-Order Stable Model Semantics

Idea 1: Treat logic programs as alternative notation for first-order formulas.

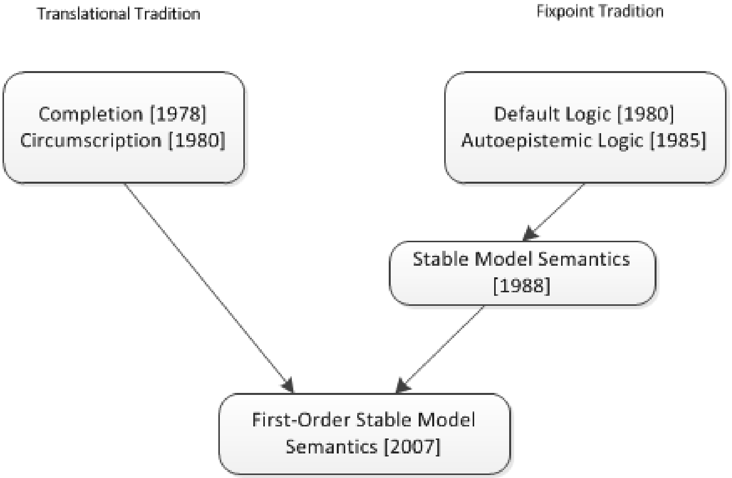
Logic program	FOL-representation
$p(X) \leftarrow \mathit{not}\ q(X), r(X)$	$\forall X(\neg q(X) \wedge r(X) \rightarrow p(X))$
$q(a)$	$\wedge q(a)$
$r(b)$	$\wedge r(b)$

Idea 2: Define the stable models of F as the models of

$$\mathrm{SM}[F; \mathbf{p}] = F \wedge (\text{2nd-order formula that enforces } \mathbf{p} \text{ to be } \textit{stable})$$

Similar to circumscription. (c.f. stability vs. minimality)

Translation vs. Fixpoint Traditions in Nonmonotonic Reasoning



Circumscription

The models of $\mathrm{CIRC}[F; \mathbf{p}]$ are the models of F that are minimal on $\mathbf{p}$. Formally,

$$\mathrm{CIRC}[F; \mathbf{p}] = F \wedge \neg \exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F(\mathbf{u}))$$

- $\mathbf{u}$: a list of distinct predicate variables similar to $\mathbf{p}$;
- $\mathbf{u} < \mathbf{p}$: a formula that expresses that $\mathbf{u}$ is strictly stronger than $\mathbf{p}$:
  - $u \leq p$ is defined as $\forall \mathbf{x}(u(\mathbf{x}) \rightarrow p(\mathbf{x}))$
  - $u = p$ is defined as $\forall \mathbf{x}(u(\mathbf{x}) \leftrightarrow p(\mathbf{x}))$
  - $\mathbf{u} < \mathbf{p}$ is defined as $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{u} = \mathbf{p})$
- $F(\mathbf{u})$ is obtained from F by replacing all occurrences of $\mathbf{p}$ with $\mathbf{u}$.

First-Order Stable Model Semantics

The stable models of a first-order sentence F relative to a list $\mathbf{p}$ of (intensional) predicate constants are the models of

$$\mathrm{SM}[F;\mathbf{p}] = F \land \neg\exists\mathbf{u}(\mathbf{u} < \mathbf{p} \land F^*(\mathbf{u}))$$

$F^*(\mathbf{u})$ is defined as:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ if $p_i \in \mathbf{p}$
- for other atomic formula F, $\;F^* = F$
- $(\neg G)^* = \neg G^* \land \neg G$;
- $(G \odot H)^* = (G^* \odot H^*) \land (G \odot H) \qquad (\odot \in \{\land, \lor, \rightarrow\})$
- $(QxG)^* = QxG^* \land QxG \qquad (Q \in \{\forall, \exists\})$

If we drop the red parts, $F^*(\mathbf{u})$ becomes the same as $F(\mathbf{u})$, so SM becomes exactly the definition of CIRC.

First-Order Stable Model Semantics

The stable models of a first-order sentence F relative to a list $\mathbf{p}$ of (intensional) predicate constants are the models of

$$\mathrm{SM}[F; \mathbf{p}] = F \wedge \neg\exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F^*(\mathbf{u}))$$

$F^*(\mathbf{u})$ is defined as:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ if $p_i \in \mathbf{p}$
- for other atomic formula F, $\;F^* = F$
- $(\neg G)^* = \neg G^* \wedge \neg G$;
- $(G \odot H)^* = (G^* \odot H^*) \wedge (G \odot H) \qquad (\odot \in \{\wedge, \vee, \rightarrow\})$
- $(QxG)^* = QxG^* \wedge QxG \qquad (Q \in \{\forall, \exists\})$

If we drop the red parts, $F^*(\mathbf{u})$ becomes the same as $F(\mathbf{u})$, so SM becomes exactly the definition of CIRC.

Observe that $\mathbf{u} < \mathbf{p} \wedge F^*(\mathbf{u}) \rightarrow F$ is logically valid.

First-Order Stable Model Semantics

The stable models of a first-order sentence F relative to a list $\mathbf{p}$ of (intensional) predicate constants are the models of the second-order formula

$$\mathrm{SM}[F;\mathbf{p}] = F \wedge \neg\exists\mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F^*(\mathbf{u}))$$

$F^*(\mathbf{u})$ can be simplified as:

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ if $p_i \in \mathbf{p}$
- for other atomic formula F, $\;F^* = F$
- $(\neg G)^* = \neg G$
- $(G \odot H)^* = (G^* \odot H^*) \qquad (\odot \in \{\wedge, \vee\})$
- $(G \to H)^* = (G^* \to H^*) \wedge (G \to H)$
- $(QxG)^* = QxG^* \qquad (Q \in \{\forall, \exists\})$

$\wedge$, $\vee$, $\forall$, $\exists$ are monotone

$\quad$(e.g., $\mathbf{u} < \mathbf{p} \wedge (F^*(\mathbf{u}) \wedge G^*(\mathbf{u})) \to (F \wedge G)$)

$\neg$ is anti-monotone

$\quad$(e.g., $\mathbf{u} < \mathbf{p} \wedge \neg F \to \neg F^*(\mathbf{u})$)

$\to$ is neither monotone nor anti-monotone

Example

$$\mathrm{SM}[F; \mathbf{p}] = F \wedge \neg\exists \mathbf{u}(\mathbf{u} < \mathbf{p} \wedge F^*(\mathbf{u}))$$

$$F = p(a) \wedge \forall x(p(x) \wedge \neg r(x) \rightarrow q(x))$$

$$F^*(u, v, w) = \quad u(a) \wedge \forall x(u(x) \wedge \neg r(x) \rightarrow v(x)) \wedge (p(x) \wedge \neg r(x) \rightarrow q(x))$$

$\mathrm{SM}[F; p, q, r]$ is equivalent to

$$F \wedge \neg\exists uvw((u, v, w) < (p, q, r) \wedge u(a) \wedge \forall x(u(x) \wedge \neg r(x) \rightarrow v(x))$$

which can be written in FOL

$$\forall x(p(x) \leftrightarrow x = a) \wedge \forall x(q(x) \leftrightarrow p(x) \wedge \neg r(x)) \wedge \forall x(r(x) \leftrightarrow \bot)$$

c.f. $\mathrm{CIRC}[F; p, q, r]$ is equivalent to

$$F \wedge \neg\exists uvw((u, v, w) < (p, q, r) \wedge u(a) \wedge \forall x(u(x) \wedge \neg w(x) \rightarrow v(x))$$

Relating SM to Traditional ASP

Theorem

The stable models of a logic program Π according to the 1988 definition are precisely the Herbrand models of $\mathrm{SM}[\Pi; pr(\Pi)]$.

Example

$\{p(a), q(a)\}$ is the unique

- stable model of $\left\{\begin{array}{l} p(a) \\ q(x) \leftarrow p(x), not\ r(x) \end{array}\right.$ under the 1988 definition
- Herbrand model of $\mathrm{SM}[p(a) \wedge \forall x(p(x) \wedge \neg r(x) \rightarrow q(x)); p, q, r]$.

System F2LP [LP09] for Computing Herbrand Stable Models of First-Order Formulas

The input languages of ASP solvers do not allow complex formulas.

F2LP is a front-end to ASP solvers that turns first-order formulas into logic program syntax.

- `f2lp [input-program] | clingo`

The F2LP rule

```
t(X) <- v(X) & not ?[Y]:e(X,Y)
```

describes the set t of terminal vertices (the symbol `?` represents the existential quantifier).

Stable Models and Circumscription

Theorem

The stable model semantics and circumscription coincide on the class of canonical formulas [LP12b].

In other words, minimal models and stable models coincide on canonical formulas.

The theorem allows us to reformulate the Event Calculus, the Situation Calculus, and Temporal Action Logics in ASP, and use ASP solvers to compute them

- Event Calculus in ASP [KLP09, LP12b]
- Situation Calculus in ASP [LP10, LP12b]
- Integrating rules and ontologies in FOSM [LP11]
- Temporal Action Logics in ASP [LP12a]

Event Calculus in Terms of ASP

Theorem

$$
\begin{array}{ll}
& \mathrm{CIRC}[\Sigma; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge \mathrm{CIRC}[\Delta; \mathit{Happens}] \wedge F \\
\iff & \quad \{\textit{By the theorem on canonical formulas}\} \\
& \mathrm{SM}[\Sigma; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge \mathrm{SM}[\Delta; \mathit{Happens}] \wedge F \\
\iff & \quad \{\textit{By the splitting theorem}\} \\
& \mathrm{SM}[\Sigma \wedge \Delta \wedge F; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}, \mathit{Happens}]
\end{array}
$$

F2LP turns $\Sigma \wedge \Delta \wedge F$ into the input language of ASP solvers.

http://reasoning.eas.asu.edu/ecasp

http://decreasoner.sourceforge.net/csr/ecas/

ASP-based vs. SAT-based Approach

- DEC reasoner is based on the reduction of circumscription to completion. Able to solve 11 out of 14 benchmark problems.
- ECASP can handle the *full* version of the event calculus (modulo grounding). Able to solve all 14 problems.
- For example, the following axiom cannot be handled by the DEC reasoner, but can be done by the ASP approach.

$$
\begin{aligned}
&\mathit{HoldsAt}(\mathit{HasBananas}, t)\\
&\wedge \mathit{Initiates}(e, \mathit{At}(\mathit{Monkey}, l), t) \rightarrow \mathit{Initiates}(e, \mathit{At}(\mathit{Bananas}, l), t)
\end{aligned}
$$

- ECASP computes faster.

Experiments

Problem (max. time)	DEC reasoner	ECASP w/ LPARSE + CMODELS	ECASP w/ GRINGO + CLASP	ECASP W CLINGO
Falling w/ AntiTraj (15)	270.2 (269.3+0.9) A:416/C:3056	0.74 (0.66+0.08) A:5757/R:10480/C:0	0.10 (0.08+0.02) A:4121/R:7820	0.08
Falling w/ Events (25)	107.70 (107.50+0.20) A:1092/C:12351	34.77 (30.99+3.78) A:1197/R:390319/C:1393	2.90 (2.01+0.89) A:139995/R:208282	2.32
HotAir Balloon (15)	61.10 (61.10+0.00) A:288/C:1163	0.19 (0.16+0.03) A:489/R:2958/C:678	0.04 (0.03+0.01) A:1137/R:1909	0.03
Telephone1 (40)	18.00 (17.50+0.50) A:5419/C:41750	1.70 (1.51+0.19) A:23978/R:30005/C:0	0.31 (0.26+0.05) A:21333/R:27201	0.25

A: number of atoms, C: number of clauses, R: number of ground rules

DEC reasoner and CMODELS used the same SAT solver RELSAT.

$$
\begin{array}{ll}
 & \mathrm{CIRC}[\Gamma_{occ}; \mathit{Occurs}] \wedge \mathrm{CIRC}[\Gamma_{depc} \wedge \Gamma_{acs}; \mathit{Occlude}] \;\; \wedge \Gamma_{rest} \\
\Longleftrightarrow & \quad \{\text{By the theorem on canonical formulas}\} \\
 & \mathrm{SM}[\Gamma_{occ}; \mathit{Occurs}] \wedge \mathrm{SM}[\Gamma_{depc} \wedge \Gamma_{acs}; \mathit{Occlude}] \;\; \wedge \Gamma_{rest} \\
\Longleftrightarrow & \quad \{\text{By the symmetric splitting theorem}\} \\
 & \mathrm{SM}[\Gamma_{occ} \wedge \Gamma_{depc} \wedge \Gamma_{acs} \wedge \Gamma_{rest}; \mathit{Occurs}, \mathit{Occlude}] \; .
\end{array}
$$

System F2LP turns the formula $\Gamma_{occ} \wedge \Gamma_{depc} \wedge \Gamma_{acs} \wedge \Gamma_{rest}$ into the input language of ASP solvers.

Integrating TAL with Ontologies

Integrating TAL with description logics is straightforward as they are both based on classical logic (**c.f.** integrating rules and ontologies).

Example

A patient is suffering from gastritis and complains of abdominal pain. Is aspirin recommended?

- In OWL ontology (e.g., National Drug File):
  - Aspirin may treat pain
  - Gastrointestinal bleeding is a contraindication for administering aspirin.
- In TAL: Gastritis causes gastrointestinal bleeding.

Answer: No, since gastrointestinal bleeding is a contraindication.

Representing the Example in TAL (Part)

Gastritis usually causes gastrointestinal bleeding:

$$\textbf{dep1}\ \forall t([t]\ Cond(Gastritis) \wedge \neg Ab_2(Gastritis) \rightarrow R([t+1]\ Cond(GastrointestinalBleeding))).$$

Domain rules for drug recommendation:

$$\textbf{acs1}\ [t_1, t_2]\ Administer(d) \rightarrow \Big([t_1]\ Cond(c) \wedge \neg Ab_1(d, c) \wedge (\mathtt{MAY\_TREAT}(d, c) \vee \mathtt{MAY\_PREVENT}(d, c)) \wedge \neg\exists c_1, t\big(t_1 \leq t \leq t_2 \wedge [t]\ Cond(c_1) \wedge \mathtt{CONTRAINDICATION}(d, c_1)\big) \rightarrow I((t_1, t_2]\ \neg Cond(c))\Big).$$

$$\textbf{acs2}\ [t_1, t_2]\ Administer(d) \rightarrow \forall t\big(t_1 \leq t \leq t_2 \wedge [t]\exists c(Cond(c) \wedge \mathtt{CONTRAINDICATION}(d, c)) \rightarrow I([t, t_2]\ SideEffect(d))\big).$$

$\mathtt{MAY\_TREAT}(Aspirin, Pain)$ and $\mathtt{CONTRAINDICATION}(Aspirin, GastrointestinalBleeding)$ can be obtained from NDF ontology by using the DLVHEX rdf plugin.

Reductive Semantics [LLP08]

- A simple, alternative approach to understanding the meaning of counting and choice in answer set programming by reducing them to first order formulas.

$$\{q(x)\} \leftarrow p(x) \quad \Rightarrow \quad \forall x(p(x) \rightarrow (q(x) \vee \neg q(x)))$$

$$r \leftarrow \#count\{x : p(x)\} \geq 2$$

$$\Rightarrow \quad (\exists xy(p(x) \wedge p(y) \wedge \neg(x = y))) \rightarrow r$$

Generalized Quantifiers

The approach doesn't cover other extensions of the stable model semantics

- Arbitrary aggregates
- (Abstract) constraints
- External atoms
- Description logic atoms

A common issue is how to incorporate "complex atoms."

$\forall$ and $\exists$ are just two instances of a much more general concept of quantifiers [Mos57]. A generalized quantifier (GQ) can represent any relation over relations.

We extended FOSM to allow generalized quantifiers and showed that all the extensions above can be viewed in terms of generalized quantifiers under FOSM [LM12a, LM12b, LM12c].

Stable Models of Formulas with Intensional Functions

Leaking Container Example

Describe a water tank that has a leak but that can be refilled to the maximum amount, say 10, with the action *FillUp*.

Functional Stable Model Semantics (FSM) [Bartholomew and Lee, 2012]

Allows for assigning default values to non-Herbrand functions, which is useful for expressing inertia and default behaviors of systems.

Leaking Container Example

$$\begin{array}{rcl} \{Amount_1 = x\} & \leftarrow & Amount_0 = x+1 \\ Amount_1 = 10 & \leftarrow & FillUp\ . \end{array}$$

$\{F\}$ is a choice rule standing for $F \vee \neg F$

- $I_1 = \{FillUp = \mathrm{F}, Amount_0 = 6, Amount_1 = 5\}$:
  I_1 is a stable model of F (relative to $Amount_1$) as well as a model.
- $I_2 = \{FillUp = \mathrm{F}, Amount_0 = 6, Amount_1 = 8\}$:
  I_2 is a model of F but not a stable model.
- $I_3 = \{FillUp = \mathrm{T}, Amount_0 = 6, Amount_1 = 10\}$:
  I_3 is a model of F as well as a stable model of F.

Infinitary Ground Formula w.r.t. an Interpretation

Since the universe may be infinite, grounding a first-order sentence F relative to an interpretation I (denoted $gr_I[F]$) may introduce infinite conjunctions and disjunctions.

Leaking Container Example. $gr_I[F]$ is

$$
\begin{array}{rcl}
\{Amount_1 = 0\} & \leftarrow & Amount_0 = 0+1 \\
\{Amount_1 = 1\} & \leftarrow & Amount_0 = 1+1 \\
 & \dots & \\
Amount_1 = 10 & \leftarrow & FillUp
\end{array}
$$

Reduct-based Definition of FSM

For any two interpretations I, J of the same signature and any list $\mathbf{c}$ of distinct predicate and function constants, we write $J <^{\mathbf{c}} I$ if

- J and I have the same universe and agree on all constants not in $\mathbf{c}$;
- $p^J \subseteq p^I$ for all predicate constants p in $\mathbf{c}$; and
- J and I do not agree on $\mathbf{c}$.

The reduct $F^{\underline{I}}$ of an infinitary ground formula F relative to an interpretation I is the formula obtained from F by replacing every maximal subformula that is not satisfied by I with $\bot$.

I is a stable model of F relative to $\mathbf{c}$ if

- I satisfies F, and
- every interpretation J such that $J <^{\mathbf{c}} I$ does not satisfy $(gr_I[F])^{\underline{I}}$.

Leaking Container Example

$I_1 = \{\mathit{FillUp}=\mathrm{F}, \mathit{Amount}_0=6, \mathit{Amount}_1=5\} \models \mathrm{SM}[F; \mathit{Amount}_1]$

$$
\begin{array}{lrcl}
gr_{I_1}(F): & \mathit{Amount}_1=0 \lor \neg(\mathit{Amount}_1=0) & \leftarrow & \mathit{Amount}_0=0+1 \\
 & & \cdots & \\
 & \mathit{Amount}_1=5 \lor \neg(\mathit{Amount}_1=5) & \leftarrow & \mathit{Amount}_0=5+1 \\
 & & \cdots & \\
 & \mathit{Amount}_1=10 & \leftarrow & \mathit{FillUp} \\
\hline
(gr_{I_1}[F])^{\underline{I_1}}: & \bot \lor \neg\bot & \leftarrow & \bot \\
 & & \cdots & \\
 & \mathit{Amount}_1=5 \lor \bot & \leftarrow & \mathit{Amount}_0=5+1 \\
 & & \cdots & \\
 & \bot & \leftarrow & \bot
\end{array}
$$

No J such that $J <^{\mathit{Amount}_1} I_1$ satisfies the reduct.

Leaking Container Example

$I_2 = \{FillUp{=}\mathrm{F}, Amount_0{=}6, Amount_1{=}8\} \not\models \mathrm{SM}[F; Amount_1]$

$$
\begin{array}{lrcl}
gr_{I_2}(F): & Amount_1{=}0 \vee \neg(Amount_1{=}0) & \leftarrow & Amount_0{=}0{+}1 \\
 & & \cdots & \\
 & Amount_1{=}5 \vee \neg(Amount_1{=}5) & \leftarrow & Amount_0{=}5{+}1 \\
 & & \cdots & \\
 & Amount_1{=}10 & \leftarrow & FillUp \\
\hline
(gr_{I_2}[F])^{\underline{I_2}}: & \bot \vee \neg\bot & \leftarrow & \bot \\
 & & \cdots & \\
 & \bot \vee \neg\bot & \leftarrow & Amount_0{=}5{+}1 \\
 & & \cdots & \\
 & \bot & \leftarrow & \bot
\end{array}
$$

I_2 satisfies the reduct, but there are also other interpretations J such that $J <^{Amount_1} I_2$ that satisfy the reduct.

Leaking Container Example

$I_3 = \{FillUp = \mathrm{T}, Amount_0 = 6, Amount_1 = 10\} \models \mathrm{SM}[F; Amount_1]$

$$
\begin{array}{llcl}
gr_{I_3}(F): & Amount_1 = 0 \vee \neg(Amount_1 = 0) & \leftarrow & Amount_0 = 0+1 \\
& & \cdots & \\
& Amount_1 = 5 \vee \neg(Amount_1 = 5) & \leftarrow & Amount_0 = 5+1 \\
& & \cdots & \\
& Amount_1 = 10 & \leftarrow & FillUp \\
\hline
(gr_{I_3}[F])^{\underline{I_3}}: & \bot \vee \neg\bot & \leftarrow & \bot \\
& & \cdots & \\
& \bot \vee \neg\bot & \leftarrow & Amount_0 = 5+1 \\
& & \cdots & \\
& Amount_1 = 10 & \leftarrow & FillUp
\end{array}
$$

No J such that $J <^{Amount_1} I_3$ satisfies the reduct.

FSM in Terms of SOL

$\mathbf{c}$ is a list of predicate and function constants called intensional.

$\mathbf{u}$ is a list of predicate and function variables corresponding to $\mathbf{c}$.

$\mathrm{SM}[F;\ \mathbf{c}]$ is defined as

$$F \wedge \neg\exists\mathbf{u}(\mathbf{u} < \mathbf{c} \wedge F^*(\mathbf{u}))$$

- For predicate symbols (variables or constants) u and c
  - $u \leq c$ is defined as $\forall\mathbf{x}(u(\mathbf{x}) \rightarrow c(\mathbf{x}))$
  - $u = c$ is defined as $\forall\mathbf{x}(u(\mathbf{x}) \leftrightarrow c(\mathbf{x}))$

- For function symbols u and c,
  - $u = c$ is defined as $\forall\mathbf{x}(u(\mathbf{x}) = c(\mathbf{x}))$

- $\mathbf{u} < \mathbf{c}$ is defined as $(\mathbf{u}^{pred} \leq \mathbf{c}^{pred}) \wedge \neg(\mathbf{u} = \mathbf{c})$

FSM in Terms of SOL

The stable models of a first-order sentence F relative to a list of distinct predicate and function constants $\mathbf{c}$ are the models of the second-order formula

$$\mathrm{SM}[F;\ \mathbf{c}] = F \wedge \neg\exists\mathbf{u}(\mathbf{u} < \mathbf{c} \wedge F^*(\mathbf{u}))$$

where $F^*(\mathbf{u})$ is defined as:

- when F is an atomic formula, F^* is $F(\mathbf{u}) \wedge F$;
- $(G \wedge H)^* = G^* \wedge H^*$; $\quad (G \vee H)^* = G^* \vee H^*$;
  $(G \rightarrow H)^* = (G^* \rightarrow H^*) \wedge (G \rightarrow H)$;
- $(\forall x G)^* = \forall x G^*$; $\quad (\exists x F)^* = \exists x F^*$.

Blocks World in FSM

$$
\begin{array}{rcl}
\bot & \leftarrow & Loc(b_1, t) = b \wedge Loc(b_2, t) = b \wedge (b_1 \neq b_2) \\
Loc(b, t+1) = l & \leftarrow & Move(b, l, t) \\
\bot & \leftarrow & Move(b, l, t) \wedge Loc(b_1, t) = b \\
\bot & \leftarrow & Move(b, b_1, t) \wedge Move(b_1, l, t) \\
\{Loc(b, 0) = l\} & & \\
\{Move(b, l, t)\} & & \\
\{Loc(b, t+1) = l\} & \leftarrow & Loc(b, t) = l \ .
\end{array}
$$

The last rule is a default formula that describes the commonsense law of inertia.

Blocks World : Eliminating Function *Loc*

For the class of **c**-plain formulas, intensional functions can be eliminated in favor of intensional predicates.

$$
\begin{array}{rcl}
\bot & \leftarrow & Loc(b_1, b, t) \land Loc(b_2, b, t) \land \neg(b_1 = b_2) \\
Loc(b, l, t+1) & \leftarrow & Move(b, l, t) \\
\bot & \leftarrow & Move(b, l, t) \land Loc(b_1, b, t) \\
\bot & \leftarrow & Move(b, b_1, t) \land Move(b_1, l, t) \\
\{Loc(b, l, 0)\} & & \\
\{Move(b, l, t)\} & & \\
\{Loc(b, l, t+1)\} & \leftarrow & Loc(b, l, t) \\
 & & \\
\bot & \leftarrow & Loc(b, l, t) \land Loc(b, l_1, t) \land \neg(l = l_1) \\
\bot & \leftarrow & \neg\exists l\ Loc(b, l, t)
\end{array}
$$

Blocks World: In the Language of ASP Solvers

```
% every block is a location
location(B) :- block(B).

% the table is a location
location(table).

:- 2{loc(BB,B,ST): block(BB)}.

loc(B,L,T+1) :- move(B,L,T).

% preconditions
:- move(B,L,T), loc(B1,B,T).
:- move(B,B1,T), move(B1,L,T).
```

```
{loc(B,L,0)}.

{move(B,L,T)}.

{loc(B,L,T+1)} :- loc(B,L,T).

% uniqueness constraint
:- 2{loc(B,LL,ST): location(LL)}.

% existence constraint
:- {loc(B,LL,ST): location(LL)}0.
```

Answer Set Programming Modulo Theories

Answer Set Programming Modulo Theories (ASPMT)

- Defined as FSM with the fixed interpretation for the background signature.

  Let σ^{bg} be the (many-sorted) signature of a background theory bg, and let J^{bg} be the (fixed) interpretation of σ^{bg}. Let σ be a signature that is disjoint from the background signature σ^{bg}.

  An interpretation I of σ is a model of an SMT sentence F w.r.t. the background theory bg, denoted by $I \models_{bg} F$, if $I \cup J^{bg}$ satisfies F.

  I is a stable model of F relative to $\mathbf{c}$ (w.r.t. background theory σ^{bg}) if $I \models_{bg} \mathrm{SM}[F; \mathbf{c}]$.

Completion : Turning ASPMT to SMT

Theorem

For any sentence F in Clark normal form that is tight on **c**, *an interpretation I that satisfies $\exists xy(x \neq y)$ is a stable model F iff I is a model of the completion of F.*

Leaking Container Example, Continued.

$$\begin{array}{rcl} \{Amount_1 = x\} & \leftarrow & Amount_0 = x+1 \\ Amount_1 = 10 & \leftarrow & FillUp\ . \end{array}$$

can be rewritten as

$$Amount_1 = x \quad \leftarrow \quad (\neg\neg(Amount_1 = x) \wedge Amount_0 = x+1) \vee \ (x = 10 \wedge FillUp)$$

and completion turns it into

$$Amount_1 = x \quad \leftrightarrow \quad (\neg\neg(Amount_1 = x) \wedge Amount_0 = x+1) \vee \ (x = 10 \wedge FillUp).$$

The formula can be written without mentioning the variable x:

$$\big((Amount_0 = Amount_1 + 1) \vee (Amount_1 = 10 \wedge FillUp)\big) \wedge (FillUp \rightarrow Amount_1 = 10)$$

In the language of SMT solver iSAT, this formula can be represented as

```
(Amt' + 1 = Amt) or (Amt' = 10 and FillUp);
FillUp -> Amt' = 10;
```

In the language of SMT solver Z3, this formula can be represented as

```
(assert (or (= (+ Amt1 1) Amt0) (and (= Amt1 10) FillUp)))
(assert (=> FillUp0 (= Amt1 10)))
```

Gears World Example

$$
\begin{array}{rcl}
M1Speed_t = x & \leftarrow & M1Speed_{t-1} = x - 1 \wedge IncreaseM1_{t-1} \\
M1Speed_t = x & \leftarrow & M1Speed_{t-1} = x \wedge \neg\neg(M1Speed_t = x) \\
 & \ldots & \\
Gear1Speed_t = \; x & \leftarrow & Higher_t = x \wedge Connected_t .
\end{array}
$$

Instance Size	ASP (Gringo+Clasp) Execution		iSAT Execution	
	Run Time (Grounding + Solving)	Atoms	Run Time	Variable
5	.02s (.02s + 0s)	3174	.03s	331
10	.3s (.3s + 0s)	10161	.19s	596
20	9.46s (4.02s + 5.11s)	36695	.79s	1126
30	42.56s (22.32s + 20.24s)	77627	2.05s	1656
50	923.74s (297.26 + 626.48s)	207706	14.35s	2716
100	out of memory		494.77s	5366

Comparison with Constraint Answer Set Solving

- CLINGCON programs [GOS09] can be viewed as a special case of ASPMT instances, which allows non-Herbrand functions, but does not allow them to be intensional.
- ASP(LC) programs by [LJN12] can be viewed similarly.
- In fact, they can be viewed even as a special case of the language from [FLL11], which FSM properly generalizes.

Reformulating $\mathcal{C}+$ in ASPMT

Language $\mathcal{C}+$ [GLL+04]

- $\mathcal{C}+$ is a formal model of parts of natural language for representing and reasoning about transition systems.
- Can represent actions with conditional and indirect effects, nondeterministic actions, and concurrently executed actions.
- Can represent multi-valued fluents, defined fluents, additive fluents, and rigid constants.
- Can represent defeasible causal laws and action attributes.
- Implemented in systems CCALC, CPLUS2ASP, COALA.

Reformulating $\mathcal{C}+$ in ASPMT

- Language $\mathcal{C}+$ is an expressive action description language but its semantics was defined in terms of propositional causal theories, which limits the language to express discrete changes only.
- By reformulating $\mathcal{C}+$ in terms of ASPMT, we can apply $\mathcal{C}+$ for reasoning about continuous changes as well, and use SMT solvers to compute the language.

causal laws	In ASPMT
caused F **if** G	$\neg\neg\ i:G \rightarrow i:F$
caused F **if** G **after** H	$\neg\neg\ (i+1):G \wedge i:H \rightarrow (i+1):F$

Representing Continuous Changes in Enhanced $\mathcal{C}+$

We distinguish between steps and real clock times. We assume the Theory of Reals as the background theory, and introduce

- *Time*: a simple fluent constant with value sort $\mathcal{R}_{\geq 0}$ (clock time);
- *Dur*: an action constant with value sort $\mathcal{R}_{\geq 0}$, which denotes the time elapsed between the two consecutive states.

We postulate:

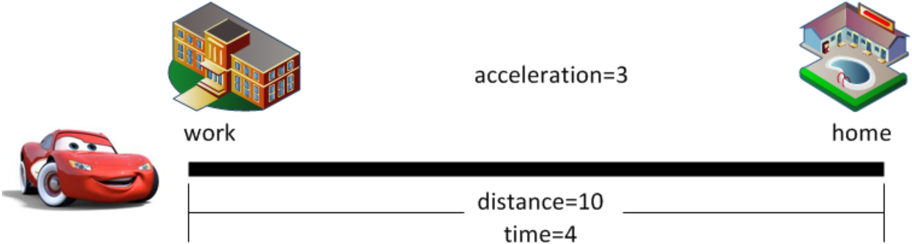
$$\begin{array}{l} \textbf{caused } \mathit{Time}=t \textbf{ if } \mathit{Time}=t \\ \textbf{caused } \mathit{Dur}=t \textbf{ if } \mathit{Dur}=t \\ \textbf{caused } \bot \textbf{ if } \neg(\mathit{Time}=t+t') \textbf{ after } \mathit{Time}=t \wedge \mathit{Dur}=t' \end{array}$$

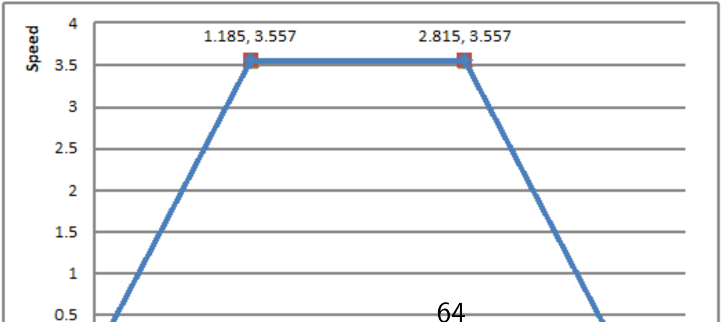
Continuous changes can be described as a function of duration using fluent dynamic laws

$$\textbf{caused } c=f(\mathbf{x},\mathbf{x}',t) \textbf{ if } \mathbf{c}'=\mathbf{x}' \textbf{ after } (\mathbf{c}=\mathbf{x}) \wedge (\mathit{Dur}=t) \wedge G$$

Planning with Continuous Time

Example: give a formal representation of the domain to generate a plan





Car Example in Enhanced $\mathcal{C}+$

Notation: d, v, v', t, t' are variables of sort $\mathcal{R}_{\geq 0}$; A, MS are real numbers.

Simple fluent constants:	Domains:
Speed, *Distance*, *Time*	$\mathcal{R}_{\geq 0}$
Action constants:	Domains:
Accelerate, *Decelerate*	Boolean
Dur	$\mathcal{R}_{\geq 0}$

Causal laws:

caused $\mathit{Speed} = v + \mathrm{A} \times t$ **after** $\mathit{Accelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t$
caused $\mathit{Speed} = v - \mathrm{A} \times t$ **after** $\mathit{Decelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t$
caused $\mathit{Distance} = d + 0.5 \times (v + v') \times t$ **if** $\mathit{Speed} = v'$
after $\mathit{Distance} = d \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t$
constraint $\mathit{Time} = t + t'$ **after** $\mathit{Time} = t \wedge \mathit{Dur} = t'$
constraint $\mathit{Speed} \leq \mathrm{MS}$

inertial *Speed*
exogenous *Time*
exogenous c for every action constant c

Turning $\mathcal{C}+$ into ASPMT and SMT

$$\mathcal{C}+ \xrightarrow{\textit{semantics}} \mathrm{ASPMT} \xrightarrow{\textit{completion}} \mathrm{SMT} \xrightarrow{\textit{eliminating variables}} \mathrm{SMT\ solvers}$$

In $\mathcal{C}+$:
caused $\mathit{Speed} = v + \mathrm{A} \times t$ **after** $\mathit{Accelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t$
caused $\mathit{Speed} = v - \mathrm{A} \times t$ **after** $\mathit{Decelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t$
caused $\mathit{Speed} = v$ **if** $\mathit{Speed} = v$ **after** $\mathit{Speed} = v$

In ASPMT:
$i{+}1 : \mathit{Speed} = x \ \leftarrow\ (x = v + \mathrm{A} \times t) \wedge\ i : (\mathit{Accelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t)$
$i{+}1 : \mathit{Speed} = x \ \leftarrow\ (x = v - \mathrm{A} \times t) \wedge\ i : (\mathit{Decelerate} \wedge \mathit{Speed} = v \wedge \mathit{Dur} = t)$
$i{+}1 : \mathit{Speed} = x \leftarrow \neg\neg(i{+}1 : \mathit{Speed} = x) \wedge i : \mathit{Speed} = x$

In SMT: The completion on $i{+}1 : \mathit{Speed}$ yields a formula that is equivalent to

$$\begin{aligned} i{+}1 : \mathit{Speed} = x \leftrightarrow\ & \big(x = (i : \mathit{Speed} + A \times i : \mathit{Dur}) \ \wedge\ i : \mathit{Accelerate}\big) \\ & \vee\ \big(x = (i : \mathit{Speed} - A \times i : \mathit{Dur}) \ \wedge\ i : \mathit{Decelerate}\big) \\ & \vee\ \big(i{+}1 : \mathit{Speed} = x \ \wedge\ i : \mathit{Speed} = x\big). \end{aligned}$$

In the language of SMT Solvers

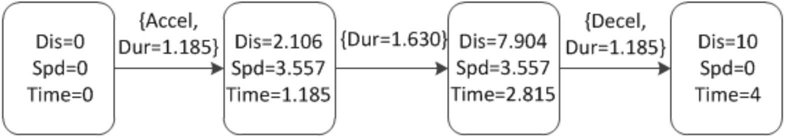
Variable x in the formula can be eliminated by equivalent transformations using equality:

$$i\!:\!Accelerate \rightarrow i\!+\!1\!:\!Speed = (i\!:\!Speed + A \times i\!:\!Dur)$$

$$i\!:\!Decelerate \rightarrow i\!+\!1\!:\!Speed = (i\!:\!Speed - A \times i\!:\!Dur)$$

$$\begin{aligned} &(i\!+\!1 : Speed = (i\!:\!Speed + A \times i\!:\!Dur) \wedge i\!:\!Accelerate) \\ \vee\ &(i\!+\!1 : Speed = (i\!:\!Speed - A \times i\!:\!Dur) \wedge i\!:\!Decelerate) \\ \vee\ &(i\!:\!Speed = i\!+\!1\!:\!Speed)\ . \end{aligned}$$

The shortest step plan found by SMT solver iSAT:
(http://isat.gforge.avacs.org)



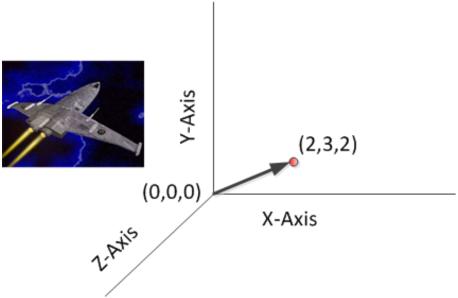
Reasoning about Indirect Effects

Indirect effects can be represented in static causal laws in $\mathcal{C}+$:

- For example, Accelerating and decelerating not only affect the speed and the distance of the car, but also indirectly affect the speed and the distance of the bag in the car.

$$\begin{array}{l}\textbf{caused } \mathit{Speed}(\mathit{Bag}) = x \textbf{ if } \mathit{Speed} = x \wedge \mathit{In}(\mathit{Bag}, \mathit{Car}) \\ \textbf{caused } \mathit{Distance}(\mathit{Bag}) = x \textbf{ if } \mathit{Distance} = x \wedge \mathit{In}(\mathit{Bag}, \mathit{Car})\ .\end{array}$$

Reasoning about Additive Fluents



Describe the cumulative effects of firing multiple jets:

- In the language of CCALC:
  $\mathit{Fire}(j)$ **increments** $\mathit{Vel}(ax)$ **by** n/Mass **if** $\mathit{Force}(j, ax) = n$
  limited to integer arithmetic.
- In enhanced $\mathcal{C}+$:
  $\mathit{Fire}(j)$ **increments** $\mathit{Vel}(ax)$ **by** $n/\mathit{Mass} \times t$ **if** $\mathit{Force}(j, ax) = n \wedge \mathit{Dur} = t$.

Even for integer domains, computing $\mathcal{C}+$ using SMT solvers was more effective than CCALC (which uses SAT solvers) and CPLUS2ASP (which uses ASP solvers).

Max Step	CCalc		CPLUS2ASP		$\mathcal{C}$+ in iSAT v1.0	
	Run Time (grounding+solving)	# of atoms / clauses	Run Time (grounding+solving)	# of atoms / rules	Run Time last/total	# of variables / clauses (bool + real)
1	0.16 (0.12+0.00)	488 / 1872	0.005 (0.005+0)	1864 / 2626	0/0	(42+53) / 182
2	0.57 (0.40+0.00)	3262 / 14238	0.033 (0.033+0)	6673 / 12035	0/0	(82+98) / 352
3	10.2 (2.62+6)	32772 / 155058	0.434 (0.234+0.2)	42778 / 92124	0/0	(122+143) / 520
4	505.86 (12.94+479)	204230 / 992838	12.546 (3.176+9.37)	228575/ 503141	0/0	(162+188) / 688
5	failed (51.10+failed)	897016 / 4410186	73.066 (15.846+57.22)	949240/ 2060834	0/0.03	(202+233) / 856
6	time out	–	3020.851 (62.381+2958.47)	3179869/ 6790167	0/0.03	(242+278) / 1024
10	time out	–	time out	–	0.03/0.09	(402+458) / 1696
50	time out	–	time out	–	0.09/1.39	(2002+2258) / 8416
100	time out	–	time out	–	0.17/5.21	(4002+4508) / 16816
200	time out	–	time out	–	0.33/21.96	(8002+9008) / 33616

The enhanced $\mathcal{C}+$ is flexible enough to represent the start-process-end model, where instantaneous actions may initiate or terminate processes.

Example: Two Taps Water Tank with Leak
TurnOn(x) **causes** *On*$(x) \wedge$ *Dur* $= 0$
TurnOff(x) **causes** *On*$(x) = \mathrm{F} \wedge$ *Dur* $= 0$

On(x) **increments** *Level* **by** $\mathrm{W}(x) \times t$ **if** *Dur* $= t$
Leaking **increments** *Level* **by** $-(\mathrm{V} \times t)$ **if** *Dur* $= t$

constraint $(\mathrm{Low} \leq \mathit{Level}) \wedge (\mathit{Level} \leq \mathrm{High})$
inertial *On*(x), *Leaking*
exogenous c for every action constant c

exogenous *Time*
constraint $\mathit{Time} = t + t'$ **after** $\mathit{Time} = t \wedge \mathit{Dur} = t'$

Conclusion

- ASPMT is a natural formalism that combines the advantages of ASP and SMT. Enhancements in ASP and SMT can be carried over to ASPMT.
- We expect that many results known between ASP and SAT can be carried over to the relationship between ASPMT and SMT. Completion is one such example.
- The enhanced C+, defined by a reduction to ASPMT, allows us to handle reasoning about hybrid systems, where discrete state changes and continuous changes coexist.
- See Related Presentations at IJCAI.
  - Functional Stable Model Semantics and Answer Set Programming Modulo Theories.
  - Answer Set Programming Modulo Theories and Reasoning about Continuous Changes.
  - Action Language BC: Preliminary Report.

Bibliography

Erdi Aker, Ahmetcan Erdogan, Esra Erdem, and Volkan Patoglu.
Causal reasoning for planning and coordination of multiple housekeeping robots.
In Proc. of LPNMR, pages 311–316, 2011.

M. Alviano, W. Faber, and N. Leone.
Disjunctive asp with functions: Decidable queries and effective computation.
TPLP, 10(4-6):497–512, 2010.

Erdi Aker, Volkan Patoglu, and Esra Erdem.
Answer set programming for reasoning with semantic knowledge in collaborative housekeeping robotics.
In Proc. of the 10'th IFAC Symposium on Robot Control (SYROCO), 2012.

Marcello Balduccini.
Representing constraint satisfaction problems in answer set programming.
In Working Notes of the Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP), 2009.

Marcello Balduccini.
Industrial-size scheduling with asp+cp.
In Proceedings of the 11th international conference on Logic programming and nonmonotonic reasoning, pages 284–296, 2011.

G. Boenn, M. Brain, M. D. Vos, and J. Fitch.
Anton: Composing logic and logic composing.
In Proc. of LPNMR, pages 542–547, 2009.

Georg Boenn, Martin Brain, Marina De Vos, and John Fitch.
Automatic music composition using answer set programming.
TPLP, 11(2–3):397–427, 2011.

Olivier Bodenreider, Zeynep H. Coban, Mahir C. Doganay, Esra Erdem, and Hilal Kosucu.
A preliminary report on answering complex queries related to drug discovery using answer set programming.
In Proc. of the 3rd Int'l Workshop on Applications of Logic Prog. to the Semantic Web and Web Services (ALPSWS), 2008.

Chitta Baral and Juraj Dzifcak.
Solving puzzles described in english by automated translation to answer set programming and learning how to do that translation.
In Proc. of KR, 2012.

Chitta Baral, Juraj Dzifcak, and Tran Cao Son.
Using answer set programming and lambda calculus to characterize natural language sentences with normatives and exceptions.
In Proc. of AAAI, pages 818–823, 2008.

D. R. Brooks, E. Erdem, S. T. Erdogan, J. W. Minett, and D. Ringe.
Inferring phylogenetic trees using answer set programming.
J. Autom. Reasoning, 39(4):471–511, 2007.

Gerhard Brewka, Thomas Eiter, and Michael Fink.
Nonmonotonic multi-context systems: A flexible approach for integrating heterogeneous knowledge sources.
In Proc. of LPNMR, pages 233–258, 2011.

Gerhard Brewka, Thomas Eiter, Michael Fink, and Antonius Weinzierl.
Managed multi-context systems.
In Proc. of IJCAI, pages 786–791, 2011.

M. Balduccini and M. Gelfond.
Diagnostic reasoning with a-prolog.
CoRR, cs.AI/0312040, 2003.

Chitta Baral, Marcos Alvarez Gonzalez, and Aaron Gottesman.
The inverse lambda calculus algorithm for typed first order logic lambda calculus and its application to translating english to fol.
In Correct Reasoning, pages 40–56, 2012.

Chitta Baral, Michael Gelfond, and J. Nelson Rushton.
Probabilistic reasoning with answer sets.
TPLP, 9(1):57–144, 2009.

Chitta Baral, Gregory Gelfond, Tran Cao Son, and Enrico Pontelli.
Using answer set programming to model multi-agent scenarios involving agents' knowledge about other's knowledge.
In Proc. of AAMAS, pages 259–266, 2010.

Michael Bartholomew and Joohyung Lee.
Stable models of formulas with intensional functions.
In Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR), pages 2–12, 2012.

Michael Bartholomew and Joohyung Lee.
Functional stable model semantics and answer set programming modulo theories.
In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 2013.
To appear.

Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski.
Preferences and nonmonotonic reasoning.
AI Magazine, 29(4):69–78, 2008.

Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski.
Preferences and nonmonotonic reasoning.
AI Magazine, 29(4):69–78, 2008.

G. Brewka.
Preferences, contexts and answer sets.
In Proc. of ICLP, page 22, 2007.

C. Baral and C. Uyan.
Declarative specification and solution of combinatorial auctions using logic programming.
In Proc. of LPNMR, pages 186–199, 2001.

D. Cakmak, E. Erdem, and H. Erdogan.
Computing weighted solutions in ASP: Representation-based method vs. search-based method.
Ann. Math. Artif. Intell., 62(3–4):219–258, 2011.

D. Calvanese, T. Eiter, and M. Ortiz.

Regular path queries in expressive description logics with nominals.
In Proc. of IJCAI, pages 714–720, 2009.

Ozan Caldiran, Kadir Haspalamutgil, Abdullah Ok, Can Palaz, Esra Erdem, and Volkan Patoglu.
Bridging the gap between high-level reasoning and low-level control.
In Proc. of LPNMR, pages 342–354, 2009.

Domenico Corapi, Daniel Sykes, Katsumi Inoue, and Alessandra Russo.
Probabilistic rule learning in nonmonotonic domains.
In Proc. of CLIMA, pages 243–258, 2011.

J. Dix, T. Eiter, M. Fink, A. Polleres, and Y. Zhang.
Monitoring agents using declarative planning.
Fundam. Inform., 57(2-4):345–370, 2003.

James P. Delgrande.
A program-level approach to revising logic programs under the answer set semantics.
TPLP, 10(4-6):565–580, 2010.

Jürgen Dix, Wolfgang Faber, and V. S. Subrahmanian.
Privacy preservation using multi-context systems and default logic.
In Correct Reasoning, pages 195–210, 2012.

J. P. Delgrande, T. Grote, and A. Hunter.
A general approach to the verification of cryptographic protocols using answer set programming.
In Proc. of LPNMR, pages 355–367, 2009.

T. Eiter, G. Brewka, M. Dao-Tran, M. Fink, G. Ianni, and T. Krennwallner.
Combining nonmonotonic knowledge bases with external sources.
In Proc. of FroCos, pages 18–42, 2009.

Halit Erdogan, Esra Erdem, and Olivier Bodenreider.
Exploiting umls semantics for checking semantic consistency among umls concepts.
In Proc. of MedInfo, 2010.

T. Eiter, E. Erdem, H. Erdogan, and M. Fink.
Finding similar or diverse solutions in answer set programming.
In Proc. of ICLP, pages 342–356, 2009.

Esra Erdem, Yelda Erdem, Halit Erdogan, and Umut Oztok.
Finding answers and generating explanations for complex biomedical queries.
In Proc. of AAAI, pages 785–790, 2011.

E. Erdem, O. Erdem, and F. Türe.
Haplo-asp: Haplotype inference using answer set programming.
In Proc. of LPNMR, pages 573–578, 2009.

T. Eiter, W. Faber, N. Leone, and G. Pfeifer.
The diagnosis frontend of the dlv system.
AI Communications, 12(1-2):99–111, 1999.

Thomas Eiter, Michael Fink, and Peter Schüller.
Approximations for explanations of inconsistency in partially known multi-context systems.
In Proc. of LPNMR, pages 107–119, 2011.

Thomas Eiter, G.Ianni, R.Schindlauer, and H.Tompits.
Effective integration of declarative rules with external evaluations for Semantic-Web reasoning.
In Proc. of ESWC, pages 273–287, 2006.

Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran.
Aspartix: Implementing argumentation frameworks using answer-set programming.
In Proc. of ICLP, pages 734–738, 2008.

Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran.
Answer-set programming encodings for argumentation frameworks.
Argument and Computation, 1(2):147–177, 2010.

Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras.
Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation.
In Proc. of the 2011 IEEE International Conference on Robotics and Automation (ICRA 2011), pages 4575–4581, 2011.

Esra Erdem, Kadir Haspalamutgil, Volkan Patoglu, and Tansel Uras.
Causality-based planning and diagnostic reasoning for cognitive factories.
In Proc. of the 17'th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2012.

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.
Combining answer set programming with description logics for the semantic web.
Artificial Intelligence, 172(12-13):1495–1539, 2008.

Esra Erdem, Katsumi Inoue, Johannes Oetsch, Joerg Puehrer, Hans Tompits, and Cemal Yilmaz.
Answer-set programming as a new approach to event-sequence testing.
In Proc. of the 3rd International Conference on Advances in System Testing and Validation Lifecycle (VALID'11), 2011.

Thomas Eiter, Thomas Krennwallner, Patrik Schneider, and Guohui Xiao.
Uniform evaluation of nonmonotonic dl-programs.
In Proc. of FoIKS, pages 1–22, 2012.

E. Erdem, V. Lifschitz, and D. Ringe.
Temporal phylogenetic networks and logic programming.
Theory and Practice of Logic Programming, 6(5):539–558, 2006.

E. Erdem, V. Lifschitz, and M. F. Wong.
Wire routing and satisfiability planning.
In In Proceedings CL-2000, pages 822–836. Springer-Verlag. LNCS, 2000.

E. Erdem.
Phylo-asp: Phylogenetic systematics with answer set programming.
In Proc. of LPNMR, pages 567–572, 2009.

Esra Erdem.
Applications of answer set programming in phylogenetic systematics.
In Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, pages 415–431, 2011.

D. East and M. Truszczynski.
More on wire routing with asp.
In Proc. of ASP, 2001.

Thomas Eiter and Kewen Wang.
Semantic forgetting in answer set programming.
Artif. Intell., 172(14):1644–1672, 2008.

Esra Erdem and Reyyan Yeniterzi.
Transforming controlled natural language biomedical queries into answer set programs.
In Proc. of BioNLP, pages 117–124, 2009.

Paolo Ferraris.
Answer sets for propositional theories.
In Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pages 119–131, 2005.

Fernando Zacarias Flores, Mauricio Javier Osorio Galindo, and Edgar Fernandez Plascencia.
Updates under pstable.
Engineering Letters, 15(2):311–315, 2007.

Paolo Ferraris and Vladimir Lifschitz.
On the stable model semantics of first-order formulas with aggregates.
In Proceedings of the 2010 Workshop on Nonmonotonic Reasoning, 2010.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz.
A new perspective on stable models.
In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pages 372–379, 2007.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz.
Stable models and circumscription.
Artificial Intelligence, 175:236–263, 2011.

Wolfgang Faber, Nicola Leone, and Gerald Pfeifer.
Recursive aggregates in disjunctive logic programs: Semantics and complexity.
In Proceedings of European Conference on Logics in Artificial Intelligence (JELIA), 2004.

R. Finkel, V. W. Marek, and M. Truszczynski.
Constraint lingo: A program for solving logic puzzles and other tabular constraint problems, 2002.

Sarah Alice Gaggl.
Towards a general argumentation system based on answer-set programming.
In ICLP (Technical Communications), pages 265–269, 2010.

M. Gebser, C. Guziolowski, M. Ivanchev, T. Schaub, A. Siegel, S. Thiele, and P. Veber.
Repair and prediction (under inconsistency) in large biological networks with answer set programming.
In Proc. of KR, 2010.

M. Gebser, R. Kaminski, and T. Schaub.
aspcud: A Linux package configuration tool based on answer set programming.
In Proc. of LoCoCo, pages 12–25, 2011.

Martin Gebser, Roland Kaufmann, and Torsten Schaub.
Gearing up for effective asp planning.
In Correct Reasoning, pages 296–310, 2012.

Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner.
Nonmonotonic causal theories.
Artificial Intelligence, 153(1–2):49–104, 2004.

Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov.
What are the limitations of the situation calculus?
In Robert Boyer, editor, Automated Reasoning: Essays in Honor of Woody Bledsoe, pages 167–179. Kluwer, 1991.

M. Gebser, M. Ostrowski, and T. Schaub.
Constraint answer set solving.
In Proceedings of International Conference on Logic Programming (ICLP), pages 235–249, 2009.

M. Gebser, T. Schaub, S. Thiele, and P. Veber.
Detecting inconsistencies in large biological networks with answer set programming.
Theory and Practice of Logic Programming, 11(2):1–38, 2011.

K. Heljanko and I. Niemela.
Bounded LTL model checking with stable models.
In Proc. of LPNMR, pages 200–212, 2003.

K. Inoue and C. Sakama.
Abductive framework for nonmonotonic theory change.
In IJCAI, pages 204–210, 1995.

Tomi Janhunen, Guohua Liu, and Ilkka Niemel.
Tight integration of non-ground answer set programming and satisfiability modulo theories.
In Working notes of the 1st Workshop on Grounding and Transformations for Theories with Variables, 2011.

Tae-Won Kim, Joohyung Lee, and Ravi Palla.
Circumscriptive event calculus as answer set programming.
In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pages 823–829, 2009.

N. Leone, G. Greco, G. Ianni, V. Lio, G. Terracina, T. Eiter, W. Faber, M. Fink, G. Gottlob, R. Rosati, D. Lembo, M. Lenzerini, M. Ruzzi, E. Kalka, B. Nowicki, and W. Staniszkis.
The infomix system for advanced integration of incomplete and inconsistent data.
In Proc. of SIGMOD, pages 915–917, 2005.

V. Lifschitz.
Answer set programming and plan generation.
Artif. Intell., 138(1-2):39–54, 2002.

Guohua Liu, Tomi Janhunen, and Ilkka Niemelä.
Answer set programming via mixed integer programming.
In Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR), pages 32–42, 2012.

Joohyung Lee, Vladimir Lifschitz, and Ravi Palla.
A reductive semantics for counting and choice in answer set programming.
In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pages 472–479, 2008.

Joohyung Lee and Yunsong Meng.
On reductive semantics of aggregates in answer set programming.
In Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pages 182–195, 2009.

Joohyung Lee and Yunsong Meng.
Stable models of formulas with generalized quantifiers.
In Proceedings of International Workshop on Nonmonotonic Reasoning (NMR), 2012.
http://peace.eas.asu.edu/joolee/papers/smgq-nmr.pdf.

Joohyung Lee and Yunsong Meng.
Stable models of formulas with generalized quantifiers (preliminary report).
In Technical Communications of the 28th International Conference on Logic Programming, pages 61–71, 2012.

Joohyung Lee and Yunsong Meng.
Two new definitions of stable models of logic programs with generalized quantifiers.
In Working Notes of the 5th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP), 2012.

Joohyung Lee and Ravi Palla.
System F2LP – computing answer sets of first-order formulas.
In Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pages 515–521, 2009.

Joohyung Lee and Ravi Palla.
Situation calculus as answer set programming.
In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pages 309–314, 2010.

Joohyung Lee and Ravi Palla.
Integrating rules and ontologies in the first-order stable model semantics (preliminary report).
In Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pages 248–253, 2011.

Joohyung Lee and Ravi Palla.
Reformulating temporal action logics in answer set programming.
In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pages 786–792, 2012.

Joohyung Lee and Ravi Palla.
Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming.
Journal of Artificial Inteligence Research (JAIR), 43:571–620, 2012.

Yuliya Lierler and Peter Schüller.
Parsing combinatory categorial grammar via planning in answer set programming.
In Correct Reasoning, pages 436–453, 2012.

A. Mileo, D. Merico, and R. Bisiani.
Wireless sensor networks supporting context-aware reasoning in assisted living.
In Proc. of PETRA, pages 1–2, 2008.

A. Mileo, D. Merico, and R. Bisiani.
Non-monotonic reasoning supporting wireless sensor networks for intelligent monitoring: The sindi system.
In Proc. of LPNMR, pages 585–590, 2009.

A. Mostowski.
On a Generalization of Quantifiers.
Fundamenta Mathematicae, 44:12–35, 1957.

A. Mileo, T. Schaub, D. Merico, and R. Bisiani.
Knowledge-based multi-criteria optimization to support indoor positioning.
AMAI, 62(3–4):345–370, 2011.

M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry.
An a-prolog decision support system for the space shuttle.
In Proc. of PADL, pages 169–183. Springer, 2001.

Mauricio Osorio and Victor Cuevas.
Updates in answer set programming: An approach based on basic structural properties.
TPLP, 7(4):451–479, 2007.

Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe.
Well-founded and stable semantics of logic programs with aggregates.
TPLP, 7(3):301–353, 2007.

Jörg Pührer, Stijn Heymans, and Thomas Eiter.
Dealing with inconsistency when combining ontologies and rules using dl-programs.
In Proc. of ESWC (1), pages 183–197, 2010.

Enrico Pontelli, Tran Cao Son, Chitta Baral, and Gregory Gelfond.
Answer set programming and planning with knowledge and world-altering actions in multiple agent domains.
In Correct Reasoning, pages 509–526, 2012.

Francesco Ricca, Antonella Dimasi, Giovanni Grasso, Salvatore Maria Ielpa, Salvatore Iiritano, Marco Manna, and Nicola Leone.
A logic-based system for e-tourism.
Fundam. Inform., 105(1–2):35–55, 2010.

Francesco Ricca, Giovanni Grasso, Mario Alviano, Marco Manna, Vincenzino Lio, Salvatore Iiritano, and Nicola Leone.
Team-building with answer set programming in the gioia-tauro seaport.
Theory and Practice of Logic Programming, 12:361–381, 2012.

C. Sakama.
Learning by answer sets.
In Proc. of AAAI Spring Symposium:Answer Set Programming, 2001.

Chiaki Sakama.
Induction from answer sets in nonmonotonic logic programs.
ACM Trans. Comput. Log., 6(2):203–231, 2005.

Chiaki Sakama.
Dishonest reasoning by abduction.
In Proc. of IJCAI, pages 1063–1064, 2011.

Chiaki Sakama and Katsumi Inoue.
Brave induction: a logical framework for learning from incomplete information.
Machine Learning, 76(1):3–35, 2009.

Mantas Simkus.
Fusion of logic programming and description logics.
In Proc. of ICLP, pages 551–552, 2009.

T. Soininen and I. Niemelä.
Developing a declarative rule language for applications in product configuration.

In Proc. of PADL, pages 305–319, 1998.

Patrik Simons, Ilkka Niemelä, and Timo Soininen.
Extending and implementing the stable model semantics.
Artificial Intelligence, 138:181–234, 2002.

T. C. Son, E. Pontelli, and C. Sakama.
Logic programming for multiagent planning with negotiation.
In Proc. of ICLP, pages 99–114, 2009.

T. C. Son and C. Sakama.
Reasoning and planning with cooperative actions for multiagents using answer set programming.
In Proc. of DALT, pages 208–227, 2009.

T. Schaub and S. Thiele.
Metabolic network expansion with answer set programming.
In Proc. of ICLP, pages 312–326, 2009.

T. Schaub and K. Wang.
A comparative study of logic programs with preference.
In Proc. of IJCAI, pages 597–602, 2001.

Yi-Dong Shen and Kewen Wang.
Extending logic programs with description logic expressions for the semantic web.
In Proc. of ISWC (1), pages 633–648, 2011.

Tommi Syrjänen.
Including diagnostic information in configuration models.
In Proc. of Computational Logic, pages 837–851, 2000.

Luis Tari, Saadat Anwar, Shanshan Liang, Jörg Hakenberg, and Chitta Baral.
Synthesis of pharmacokinetic pathways through knowledge acquisition and automated reasoning.
In Proc. of Pacific Symposium on Biocomputing, pages 465–476, 2010.

N. Tran and C. Baral.

Reasoning about triggered actions in ansprolog and its application to molecular interactions in cells.
In Proc. of KR, pages 554–564, 2004.

F. Türe and E. Erdem.
Efficient haplotype inference with answer set programming.
In Proc. of AAAI, pages 1834–1835, 2008.

Phan Huy Tu, Tran Cao Son, Michael Gelfond, and A. Ricardo Morales.
Approximation of action theories and its application to conformant planning.
Artif. Intell., 175(1):79–119, 2011.

Juha Tiihonen, Timo Soininen, Ilkka Niemelae, and Reijo Sulonen.
A practical tool for mass-customising configurable products.
In Proc. of ICDE, pages 1290–1299, 2003.

Calvin Kai Fan Tang and Eugenia Ternovska.
Model checking abstract state machines with answer set programming.
Fundam. Inform., 77(1-2):105–141, 2007.

M. D. Vos, T. Crick, J. Padget, M. Brain, O. Cliffe, and J. Needham.
A multi-agent platform using ordered choice logic programming.
In In Declarative Agent Languages and Technologies (DALT'05), pages 72–88, 2005.

Marina De Vos and Dirk Vermeir.
Dynamic decision-making in logic programming and game theory.
In Proc. of AI, pages 36–47, 2002.

M. D. Vos and D. Vermeir.
Extending answer sets for logic programming agents.
Ann. Math. Artif. Intell., 42(1-3):103–139, 2004.

Yining Wu, Martin Caminada, and Dov M. Gabbay.
Complete extensions in argumentation coincide with 3-valued stable models in logic programming.
Studia Logica, 93(2-3):383–403, 2009.

Claudia Zepeda, José Luis Carballido, Mario Rossainz, and Mauricio Osorio.
Updates based on asp.
In Proc. of MICAI (Special Sessions), pages 63–66, 2010.