# Reformulating Temporal Action Logics in Answer Set Programming

**Joohyung Lee**
School of Computing, Informatics,
and Decision Systems Engineering
Arizona State University
Tempe, AZ, 85287, USA
joolee@asu.edu

**Ravi Palla**
Siemens Corporation
Corporate Research and Technology
Princeton, NJ, 08540, USA
ravi.palla@siemens.com

### Abstract

Temporal Action Logics (TAL) is a class of temporal logics for reasoning about actions. We present a reformulation of TAL in Answer Set Programming (ASP), and discuss some synergies it brings. First, the reformulation provides a means to compute TAL using efficient answer set solvers. Second, TAL provides a structured high-level language for ASP (possibly with constraint solving). Third, the reformulation allows us to compute integration of TAL and ontologies using answer set solvers, and we illustrate its usefulness in the healthcare domain in the context of medical expert systems.

## Introduction

Temporal Action Logics (TAL) (Doherty *et al.* 1998) is a class of logics for reasoning about action and change that are based on the *Features and Fluents* framework of Sandewall (1994). TAL supports rich features, including durative actions, durational fluents (fluents with default values), ramification constraints, and qualification constraints. Like several other action formalisms, TAL adopts circumscription (McCarthy 1980; 1986) to solve the frame and the ramification problems. On the other hand, unlike many others, actions in TAL may have variable durations, and can be inert or highly dynamic during the duration. TAL has been used in several applications, including Unmanned Aerial Vehicles (Doherty, Kvarnström, & Heintz 2009) and Question-Answering (Magnusson & Doherty 2008). Two implementations of TAL are available. VITAL[1] supports many features of TAL, including ramification constraints, qualification constraints, and durational fluents. TALplanner[2] (Kvarnström 2005) is an award-winning forward-chaining planner based on TAL.

In this paper, based on the relationship between circumscription and the stable model semantics, we show how TAL can be reformulated in Answer Set Programming (ASP), and discuss some synergies it brings. [3]



[1] http://www.ida.liu.se/∼jonkv/vital

[2] http://www.ida.liu.se/divisions/aiics/aiicssite/projects/talplanner.en.shtml

[3] In this paper, TAL refers to the language presented in (Doherty & Kvarnström 2008), which is the latest kernel of this class of logics.

First, the reformulation provides a new method of computing TAL using efficient answer set solvers. Several answer set solvers are actively being developed, including those that use SAT solvers as search engines. Thus, the improvements in ASP or SAT solvers can be immediately carried over to the computation of TAL, which will benefit the applications of TAL.

Second, in addition to action language $\mathcal{A}$ and many of its descendants (Gelfond & Lifschitz 1998) that are based on ASP, TAL can be viewed as another high-level action description language on top of ASP, which is distinct from the others in terms of a more flexible ontology of time than the ones adopted in action languages. Moreover, the reformulation enables us to view TAL as a high-level language for constraint answer set solving (Gebser, Ostrowski, & Schaub 2009) that utilizes constraint variables to represent functional fluents. Typically, functional fluents in ASP are encoded in terms of predicates as in the following rule, which represents the commonsense inertia:

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

Using functions instead, such as $\mathit{WaterLevel}(t, \mathit{tank})$, does not work under the ASP semantics for a few reasons. First, answer sets are Herbrand models: $\mathit{WaterLevel}(t{+}1, \mathit{tank}) = \mathit{WaterLevel}(t, \mathit{tank})$ is always false. Even when we turn to constraint answer set solving to model functional fluents by constraint variables, there is still a problem. A naive representation of the above in the language of the constraint ASP solver CLINGCON[4] does not correctly represent the commonsense inertia:

$$\mathit{WaterLevel}(t{+}1, \mathit{tank}) =^{\$} l \leftarrow \mathit{WaterLevel}(t, \mathit{tank}) =^{\$} l,\ \mathit{not}\ \mathit{WaterLevel}(t{+}1, \mathit{tank}) \neq^{\$} l. \quad (1)$$

The rule can be dropped without affecting the answer sets. This is due to the fact that nonmonotonicity of the (constraint) ASP semantics is related to the minimality condition for predicates, but has nothing to do with functions. On the other hand, our reformulation of TAL provides an encoding method that correctly represents functional fluents by constraint variables in the language of CLINGCON. We observe

[4] http://potassco.sourceforge.net/

that this method sometimes significantly improves computation time, in comparison with non-constraint encodings.

Third, our reformulation can be adapted so that answer set solvers such as DLVHEX[5] can be used for computing the integration of TAL and ontologies, thus enabling us to use TAL in applications where there is a need to reason about cause-effect relationships while being able to query ontologies. For example, consider a medical expert system that is required to assist physicians in diagnosis and treatment of diseases/disorders. There are several biomedical ontologies, such as the Foundational Model of Anatomy (FMA) ontology[6] (Rosse & Mejino 2003), the human disease ontology[7], and the National Drug File (NDF) ontology[8]. In order to effectively assist physicians, the expert system needs to have access to the knowledge in such ontologies. Now, consider a patient who is suffering from *gastritis* and complains of abdominal *pain*. The job of the system is to determine if *aspirin* can be recommended to the patient. Assume that the biomedical ontology being queried has the knowledge that *aspirin may treat pain* (written in RDF for instance) and *gastrointestinal bleeding is a contraindication for administering aspirin*. Further, assume that we have the knowledge that *gastritis usually causes gastrointestinal bleeding*. Now, if the system simply considers the current patient symptoms, it will conclude that aspirin can be prescribed. However, since gastritis usually causes gastrointestinal bleeding, this would not be the preferred treatment. On the other hand, if the system considers the knowledge about the effects of gastritis, then it will be able to conclude that aspirin is not recommended. Thus it is clear that the system needs to consider what conditions the current patient condition might *cause* and then check whether any of the conditions is a contraindication to prescribing the drug. So, in addition to being able to query ontologies, the system also needs to be able to reason about various *cause-effect* relationships. This is one of the main motivations for integrating nonmonotonic rules and ontologies. In contrast with those approaches, that have to integrate heterogenous semantics, TAL can be straightforwardly extended to include interfaces to ontologies. We show how the reformulation of TAL in ASP can be used to turn such *hybrid* theories into the language of DLVHEX.

## Review: Circumscription and Stable Models

We follow the definition of a stable model from (Ferraris, Lee, & Lifschitz 2011). There, stable models are defined using the "stable model operator SM" with "intensional predicates," similar to circumscription.

Let $\mathbf{p}$ be a list of distinct predicate constants $p_1, \dots, p_n$ other than equality. For any first-order formula $F$, CIRC$[F; \mathbf{p}]$ is defined as the second-order formula

$$F \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F(\mathbf{u})),$$

where $\mathbf{u}$ is a list of distinct predicate variables of the same length as $\mathbf{p}$, expression $\mathbf{u} < \mathbf{p}$ stands for a formula expressing that $\mathbf{u}$ is "stronger than" $\mathbf{p}$, as defined in (Lifschitz 1994), and $F(\mathbf{u})$ is the formula obtained from $F$ by substituting the variables $\mathbf{u}$ for the constants $\mathbf{p}$. Similarly, SM$[F; \mathbf{p}]$ is defined as the second-order formula

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

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

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any tuple $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic $F$ that does not contain members of $\mathbf{p}$;
- $(F \odot G)^* = (F^* \odot G^*), \ \odot \in \{\wedge, \vee\}$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$;
- $(QxF)^* = QxF^*, \ Q \in \{\forall, \exists\}$.

A model of $F$ (in the sense of first-order logic) is *stable* (relative to the list $\mathbf{p}$ of *intensional* predicates) if it satisfies SM$[F; \mathbf{p}]$. It is shown in (Ferraris, Lee, & Lifschitz 2011) that this definition, applied to the syntax of logic programs (by viewing them as a special case of first-order formulas) and restricted to Herbrand models, is equivalent to the traditional definition of answer sets that is based on grounding and fixpoint construction.

System F2LP ("*f*ormulas to *l*ogic *p*rograms")[9] (Lee & Palla 2009; 2012) is a front-end that allows ASP solvers to compute Herbrand stable models of "almost universal" first-order theories (Lee & Palla 2012). The translation implemented in the system first eliminates quantifiers and then turns the resulting quantifier-free formula into a logic program by flattening the nested subformulas.

Theorem 1 from (Lee & Palla 2012) tells us that for the class of "canonical formulas," the stable model semantics coincides with circumscription. We say that an occurrence of a predicate constant, or any other subexpression, in a formula $F$ is *strictly positive* if it is not contained in the antecedent of any implication. For example, in formula $\forall x(\neg p(x) \to q(x))$, the occurrence of $q$ is strictly positive (we view $\neg p(x)$ as shorthand for $p(x) \to \bot$).

We say that a formula $F$ is *canonical* relative to a list $\mathbf{p}$ of predicate constants if

- no occurrence of a predicate constant from $\mathbf{p}$ is in the antecedent of more than one implication in $F$, and
- every occurrence of a predicate constant from $\mathbf{p}$ that is in the scope of a strictly positive occurrence of $\exists$ or $\vee$ in $F$ is strictly positive in $F$.

For example, formula $\forall x(\neg p(x) \to q(x))$ is not canonical relative to $\{p, q\}$ since it does not satisfy the first clause of the definition. On the other hand, the formula is canonical relative to $\{q\}$. Formula $p(a, a) \wedge \exists x(p(x, a) \to p(b, x))$ is not canonical relative to $\{p, q\}$ since it does not satisfy the second clause.

The following theorem states that, for any canonical formula, circumscription coincides with the stable model semantics. In other words, stable models and minimal models coincide on canonical formulas. This result will be used to reformulate TAL in ASP.

[5] http://www.kr.tuwien.ac.at/research/systems/dlvhex/

[6] http://sig.biostr.washington.edu/projects/fm/

[7] http://www.obofoundry.org/

[8] http://bioportal.bioontology.org/ontologies/40402

[9] http://reasoning.eas.asu.edu/f2lp

**Theorem 1** *(Lee & Palla 2012, Theorem 1) For any canonical formula $F$ relative to* $\mathbf{p}$*,*

$$\text{CIRC}[F; \mathbf{p}] \leftrightarrow \text{SM}[F; \mathbf{p}]$$

*is logically valid.*

## Reformulating TAL in ASP

### Review: Temporal Action Logics

Due to lack of space, we refer the reader to (Doherty *et al.* 1998; Doherty & Kvarnström 2008) for the details about the syntax and the semantics of TAL.

In TAL, a scenario (or narrative) can be described in a compact surface language $\mathcal{L}$(ND), which is a high-level macro expandable language consisting of *action type specifications* (**acs**), *dependency constraints* (**dep**), *domain constraints* (**dom**), *persistence statements* (**per**), *observation statements* (**obs**), and *action occurrence statements* (**occ**). Consider the *Russian Airplane Hijack* (RAH) scenario (Doherty *et al.* 1998; Doherty & Kvarnström 2008), in which some businessmen try to board a plane while carrying certain items in their pockets. This example demonstrates the capability of TAL in handling the ramification and qualification problems. We only show the action type specifications for the actions *Put* and *Board*, and some dependency constraints. The fluent $\mathit{PossBoard}(person, airplane)$ is durational with default value *true*, and the rest of the fluents are persistent (inertial). The direct effects of actions *Put* and *Board* are represented by the following action type specifications:

**acs1** $[t_1, t_2]\mathit{Put}(person, pthing, pocket)$
$\rightarrow ([t_1]\ \mathit{Loc}(person) \hat{=} \mathit{Loc}(pthing)$
$\rightarrow R((t1, t2]\ \mathit{InPocket}(person, pthing))).$

**acs2** $[t_1, t_2]\mathit{Board}(person, airplane) \rightarrow$
$\big([t_1]\ \mathit{PossBoard}(person, airplane) \wedge$
$\mathit{Loc}(person) \hat{=} \mathit{Airport} \rightarrow$
$R([t_2]\ \mathit{Loc}(person) \hat{=} \mathit{Loc}(airplane) \wedge$
$\mathit{OnPlane}(airplane, person))\big).$

(The $R$ operator is used for *reassignment*, and ensures that the formula holds at the final timepoint in the interval.) The first statement above represents that if a person puts a thing into his/her pocket in the time interval $[t_1, t_2]$ and if the location of the person is the same as the location of the thing at $t_1$, then *InPocket* is released from inertia in the interval $(t_1, t_2]$ and the thing is in the pocket at timepoint $t_2$ (i.e., $\mathit{InPocket}(person, pthing)$ becomes true at $t_2$). The second statement is similar. One of the qualification constraints for the action *Board* is as follows:

**dep1** $[t]\mathit{InPocket}(person, gun)$
$\rightarrow I([t]\ \forall airplane(\neg \mathit{PossBoard}(person, airplane))).$

(The $I$ operator is used for *interval reassignment*, and ensures that the formula holds during the entire interval.) This dependency constraint represents that if a person has a gun in his/her pocket, then he/she is not able to board any airplane.

Reasoning about a narrative in $\mathcal{L}$(ND) is done by translating it into the base language $\mathcal{L}$(FL), which is an order-sorted classical first-order language using a linear, discrete time structure. The language uses the ternary predicates *Holds* and *Occurs*, and the binary predicate *Occlude*. The translation from $\mathcal{L}$(ND) to $\mathcal{L}$(FL) is given by the function *Trans* (Doherty & Kvarnström 2008). For instance, $\mathit{Trans}([t]f \hat{=} \omega)$ is defined as $\mathit{Holds}(t, f, \omega)$ (fluent $f$ has value $\omega$ at time $t$). Similarly, $\mathit{Trans}(R((t, t']f \hat{=} \omega))$ is defined as

$$\forall t''(t < t'' \leq t' \rightarrow \mathit{Occlude}(t'', f)) \wedge \mathit{Holds}(t', f, \omega).$$

$\mathit{Occlude}(t, f)$ represents that a persistent or durational fluent $f$ is exempt from inertia or default value assumption, respectively, at time $t$. $\mathit{Trans}([t, t']\Psi)$, where $\Psi$ is an action term, is defined as $\mathit{Occurs}(t, t', \Psi)$, which represents that action $\Psi$ occurs in the interval $[t, t']$. Following is a representation of **acs1** in the language $\mathcal{L}$(FL):

$$\begin{array}{l}\mathit{Occurs}(t_1, t_2, \mathit{Put}(person, pthing, pocket)) \rightarrow \\ \big(\exists loc\ (\mathit{Holds}(t_1, \mathit{Loc}(person), loc) \wedge \\ \quad \mathit{Holds}(t_1, \mathit{Loc}(pthing), loc)) \rightarrow \\ \quad \big(\forall t\ (t_1 < t \leq t_2 \rightarrow \\ \quad\quad \mathit{Occlude}(t, \mathit{InPocket}(person, pthing))) \wedge \\ \quad \mathit{Holds}(t_2, \mathit{InPocket}(person, pthing), \mathit{true})\big)\big).\end{array} \quad (2)$$

Consider any narrative $\mathcal{N}$ and let $\mathcal{N}_{per}, \mathcal{N}_{obs}, \mathcal{N}_{occ}, \mathcal{N}_{acs}, \mathcal{N}_{domc}$, and $\mathcal{N}_{depc}$ denote the sets of persistence statements, observation statements, action occurrence statements, action type specifications, domain constraints, and dependency constraints in $\mathcal{N}$ respectively. The TAL domain description (referred to as *preferred narrative*) $\Delta_{\mathcal{N}}$ is given by

$$\text{CIRC}[\Gamma_{occ}; \mathit{Occurs}] \wedge \text{CIRC}[\Gamma_{depc} \wedge \Gamma_{acs}; \mathit{Occlude}] \wedge \\ \wedge \Gamma_{fnd} \wedge \Gamma_{time} \wedge \Gamma_{per} \wedge \Gamma_{obs} \wedge \Gamma_{domc}, \quad (3)$$

where $\Gamma_{per}$, $\Gamma_{obs}$, $\Gamma_{occ}$, $\Gamma_{acs}$, $\Gamma_{domc}$, and $\Gamma_{depc}$ are the formulas in $\mathcal{L}$(FL) (first-order logic formulas) obtained by applying *Trans* on $\mathcal{N}_{per}$, $\mathcal{N}_{obs}$, $\mathcal{N}_{occ}$, $\mathcal{N}_{acs}$, $\mathcal{N}_{domc}$, and $\mathcal{N}_{depc}$ respectively; $\Gamma_{fnd}$ is the set of foundational axioms in $\mathcal{L}$(FL), containing unique name axioms, unique value axioms, etc., and $\Gamma_{time}$ is the axiomatization of the particular temporal structure used in TAL.

### Reformulating Temporal Action Logics in Answer Set Programming

To reformulate TAL in ASP, we first reformulate TAL in the first-order stable model semantics and then use F2LP to turn the resulting theory into ASP. The following theorem shows how TAL can be reformulated in the first-order stable model semantics. In the following, we use $\Gamma_{ncirc}$ to denote the formula $\Gamma_{fnd} \wedge \Gamma_{time} \wedge \Gamma_{per} \wedge \Gamma_{obs} \wedge \Gamma_{domc}$.

**Fact 1** *The following facts are easy to check from the syntax of TAL:*

- $\Gamma_{occ}$ *is canonical relative to Occurs;*
- $\Gamma_{depc} \wedge \Gamma_{acs}$ *is canonical relative to Occlude.*

The following theorem shows a reformulation of TAL in the first-order stable model semantics.

**Theorem 2** *For any TAL domain description (3), the following formulas are equivalent to each other:*

*(a)* CIRC$[\Gamma_{occ}; \mathit{Occurs}] \wedge$ CIRC$[\Gamma_{depc} \wedge \Gamma_{acs}; \mathit{Occlude}] \wedge \Gamma_{ncirc}$

*(b)* SM$[\Gamma_{occ}; \mathit{Occurs}] \wedge$ SM$[\Gamma_{depc} \wedge \Gamma_{acs}; \mathit{Occlude}] \wedge \Gamma_{ncirc}$

*(c)* SM$[\Gamma_{occ} \wedge \Gamma_{depc} \wedge \Gamma_{acs} \wedge \Gamma_{ncirc}; \mathit{Occurs}, \mathit{Occlude}]$ .

The equivalence between $(a)$ and $(b)$ follows from Theorem 1 and Fact 1. The equivalence between $(b)$ and $(c)$ follows from the splitting theorem from (Ferraris *et al.* 2009).

Formula

$$\Gamma_{occ} \wedge \Gamma_{depc} \wedge \Gamma_{acs} \wedge \Gamma_{ncirc} \tag{4}$$

inside SM in $(c)$ can be represented in the language of F2LP. For instance, formula (2) can be represented in the language of F2LP as follows:

```
occurs(T1,T2,put(Pe, Pt, Po)) ->
(?[Lo]:(holds(T1,loc(Pe),Lo) & holds(T1,loc(Pt),Lo))
   -> (![T]:(T > T1 & T <= T2 ->
             occlude(T,inpocket(Pe,Pt))) &
       holds(T2,inpocket(Pe,Pt),true))).
```

The translation implemented in F2LP turns the formula (4) into a logic program (in the input language of GRINGO), whose stable models are identical to the models of the formula in $(c)$ when we disregard new atoms introduced in the translation. The ability to write complex formulas in the language of F2LP greatly facilitates encoding efforts for TAL.

The following command can be used for computing the answer sets of the (complete) theory encoded in the file `RAH.f2lp` (`[N]` has to be substituted with an appropriate value representing the maximum timepoint of interest):

```
$ f2lp RAH.f2lp | gringo -c maxstep=[N] | claspD
```

## TAL in Constraint ASP

The translation *Trans* represents functional fluents using predicates. That is, it turns formulas of the form $[t]\ f \hat{=} \omega$ into $\mathit{Holds}(t, f, \omega)$. This may cause a grounding bottleneck for non-Boolean domains in our ASP-based computation. Here we present how to avoid this problem by representing functional fluents by constraint variables in the language of CLINGCON, a system that combines the ASP solver CLINGO[10] and the CP (Constraint Processing) solver GECODE[11].

Intuitively, since *Holds* is non-intensional, we can simply replace $\mathit{Holds}(t_1, f(\mathbf{x}_1), v_1)$ in (4) with $\mathit{Val}(t_1, f(\mathbf{x}_1)) = v_1$, where *Val* is a new function constant. We make the idea precise as follows. Consider a TAL narrative $\mathcal{N}$ and any subset $S$ of the set of feature symbols[12] in $\mathcal{N}$. If $F$ is formula (4), by $F_S$ we denote the formula obtained from $F$ by replacing every occurrence of $\mathit{Holds}(t_1, f(\mathbf{x}_1), v_1)$, where $f \in S$, with $\mathit{Val}(t_1, f(\mathbf{x}_1)) = v_1$.

**Theorem 3** *Let $\mathcal{N}$ be a TAL narrative, let $\Delta_{\mathcal{N}}$ be the corresponding preferred narrative (3), and let $F$ be the formula (4). Under the assumption*

$$\forall t, \mathbf{x}, v(\mathit{Holds}(t, f(\mathbf{x}), v) \leftrightarrow \mathit{Val}(t, f(\mathbf{x})) = v)$$

*for every feature symbol $f$ in a set $S$ of feature symbols,* SM$[F; \mathit{Occurs}, \mathit{Occlude}]$ *is equivalent to* SM$[F_S; \mathit{Occurs}, \mathit{Occlude}]$.

For example, consider the persistence statement

$$\textbf{per}\ \forall t\ \mathit{Per}(t, \mathit{WaterLevel}(tank))$$

which represents that $\mathit{WaterLevel}(tank)$ is an inertial fluent. This is represented in $\mathcal{L}$(FL) as follows:

$$\neg \mathit{Occlude}(t{+}1, \mathit{WaterLevel}(tank)) \rightarrow \big(\forall v\ (\mathit{Holds}(t{+}1, \mathit{WaterLevel}(tank), v) \leftrightarrow \mathit{Holds}(t, \mathit{WaterLevel}(tank), v))\big).$$

In view of Theorem 3, this can be represented as

$$\neg \mathit{Occlude}(t{+}1, \mathit{WaterLevel}(tank)) \rightarrow \mathit{Val}(t{+}1, \mathit{WaterLevel}(tank)) = \mathit{Val}(t, \mathit{WaterLevel}(tank)).$$

or in the language of CLINGCON as

```
val(T+1,waterLevel(Tank)) $== val(T,waterLevel(Tank))
     :- not occlude(T+1, waterLevel(Tank)).
```

In comparison with the rule (1), this example illustrates that the TAL style encoding allows us to represent the commonsense inertia correctly in the language of CLINGCON. Our experiments in the next section indicate that it is also computationally advantageous when fluents range over large numeric domains.

## Experimental Evaluation

VITAL ("Visualization and Implementation of Temporal Action Logics") is a tool for reasoning about actions using TAL. The tool supports rich features in TAL, including ramification constraints, qualification constraints, and durational fluents. It not only generates the models but also provides a visualization of the models, which makes it easier to verify the output and is also useful in developing TAL narratives. The primary focus in the design of the tool was on supporting a high degree of expressivity in TAL, rather than on raw performance.[13]

We compare the performance of VITAL and our ASP-based method on three scenarios. The TAL descriptions of the *Russian Airplane Hijack* (RAH) and the *Water Tank* scenarios are provided with the VITAL distribution; we encoded a TAL description of the *Zoo World*, a medium size action domain proposed by Sandewall (Akman *et al.* 2004). All experiments were done on a Pentium machine with 3.06 GHz CPU and 4GB RAM running 64-bit Linux.

Figure 1 shows the comparison of VITAL (v.2.999.910 alpha) with the following tools for the RAH and Water Tank scenarios in language $\mathcal{L}$(FL):

- F2LP (v 1.11) with GRINGO (v 3.0.3) + CLASP (v 2.0.2),[14]
- F2LP (v 1.11) with CLINGCON (v 0.1.2) (GRINGO v 2.0.2 and CLASP v 1.1.1) , and

[10] http://potassco.sourceforge.net/

[11] http://www.gecode.org/

[12] The feature symbol corresponding to a fluent $f(\mathbf{x})$ is $f$.

[13] This information was provided to us by Jonas Kvarnström in a personal communication.

[14] http://potassco.sourceforge.net/

| Problem (max. step) | VITAL | F2LP with GRINGO + CMODELS | F2LP with GRINGO + CLASP | F2LP with CLINGCON |
|---|---|---|---|---|
| RAH-proj (16) | 0.01s | 0.23s (0.08s + 0.15s) | 0.13s (0.08s + 0.05s) | 0.1s |
| RAH-proj (50) | 0.05s | 0.99s (0.32s + 0.67s) | 0.53s (0.32s + 0.21s) | 0.35s |
| RAH-plan-unit (7) | 10.56s | 0.13s (0.03s + 0.1s) | 0.08s (0.03s + 0.05s) | 0.07s |
| RAH-plan-unit (10) | $>$ 30min | 0.45s (0.1s + 0.35s) | 0.29s (0.1s + 0.19s) | 0.23s |
| RAH-plan-dur (6) | 540.59s | 0.3s (0.07s + 0.23s) | 0.11s (0.07s + 0.04s) | 0.1s |
| RAH-plan-dur (9) | $>$ 30min | 0.66s (0.13s + 0.53s) | 0.26s (0.13s + 0.13s) | 0.33s |
| WaterTank-proj-100 (16) | 0.07s | $>$ 30min [0.06s] | $>$ 30min [0.05s] | 8.92s |
| WaterTank-proj-200 (25) | 0.03s | $>$ 30min [0.14s] | $>$ 30min [0.13s] | 119.44s |
| WaterTank-proj-300 (30) | 0.01s | $>$ 30min [0.22s] | $>$ 30min [0.21s] | 591.15s |

Figure 1: Comparison of VITAL with F2LP + ASP solvers

- F2LP (v 1.11) with GRINGO (v 3.0.3) + CMODELS[15] (v 3.79) running MINISAT[16] (v 2.0 beta).

The times shown in the parentheses are "(grounding time + solving time)." The cut-off time we used for the tools to terminate and return one model is 30 minutes.

For the RAH scenario, the first two problems are projection problems and the next four are planning problems. Among the planning problems, the first two consider all actions to be of unit duration, and the next two consider actions with fixed lower and upper bounds on their durations. For each planning problem, "`max.step`" is also the minimal length of the plan found. While VITAL is in general not as efficient as TALplanner on planning problems, it allows us to represent domains, such as the RAH scenario, which involve ramification and qualification constraints, that are not allowed in TALplanner. For the Water Tank scenario, the number in the problem name is the maximum value in the domain of the numbers considered. For instance, the number domain used for "WaterTank-proj-100" is `0..100`. The times reported in square brackets is when we removed the unique value axioms ("a fluent has exactly one value at each timepoint"), and instead added axioms that uniquely define the value of each fluent at every timepoint. Such rewriting works for this particular scenario, and drastically improves the performance. For example,

```
% unique value axiom
1{holds(T,waterlev(Tank),Number):num(Number)}1.

% constraint for determining next water level
-(holds(T,waterlev(Tank),Nu1)
  & holds(T,inflow(Tank),Nu2)
  & ...
  & -holds(T+1,waterlev(Tank),Nu4)).
```

can be turned into

```
(holds(T,waterlev(Tank),Nu1)
 & holds(T,inflow(Tank),Nu2)
 & ...
 -> holds(T+1,waterlev(Tank),Nu4)).
```

[15]http://www.cs.utexas.edu/ tag/cmodels/

[16]http://minisat.se/

| Problem (max. step) | VITAL | F2LP with GRINGO+CMODELS | F2LP with GRINGO+CLASP | F2LP with CLINGCON |
|---|---|---|---|---|
| ZooWorld-proj (20) | 0.43s | 0.97s (0.39s + 0.58s) | 0.80s (0.39s + 0.41s) | 0.49s |
| ZooWorld-proj (50) | $>$ 30min | 4.85s (2.28s + 2.57s) | 4.33s (2.28s + 2.05s) | 2.03s |
| ZooWorld-post (15) | 61.63s | 0.62s (0.24s + 0.38s) | 0.51s (0.24s + 0.27s) | 0.33s |
| ZooWorld-post (15) | 17.40s | 0.59s (0.24s + 0.25s) | 0.51s (0.24s + 0.27s) | 0.33s |
| ZooWorld-post (20) | 477.4s | 0.99s (0.41s + 0.58s) | 0.82s (0.41s + 0.41s) | 0.5s |

Figure 2: VITAL vs. F2LP + ASP solvers in the Zoo World

As the results show, this simple modification has a significant impact on the times taken by GRINGO + CLASP and GRINGO + CMODELS in returning an answer set.

Figure 2 shows the comparison with VITAL for the Zoo World scenario. The first two are projection problems, and the remaining are postdiction problems.

As we can see from the results, VITAL performed quite well on (almost all of) the projection problems but is not very efficient on the planning and postdiction problems we considered. On the other hand, our ASP-based approach performed equally well on all the problems, except on those involving numeric domains, such as the Water Tank scenario. The results also clearly show the advantage of turning TAL into the language of CLINGCON for handling large numeric domains.

We also compared the performance of our method with TALplanner. Detailed results are not shown here since the version of TALplanner we obtained from the developer has several features disabled. Thus, the performance of this version is not indicative of what the planner is actually capable of. Further, the performance of TALplanner heavily depends on the quality of control rules used, while ASP-based planning typically does not make use of control rules but works with reasonable efficiency. Also, the main search method of TALplanner is depth-first search, which provides no guarantee of optimality. Having said that, the experiment showed that TALplanner was considerably faster than our ASP-based method for planning tasks. On the other hand, the plans generated by TALplanner were often much longer than the minimal length plans generated by our ASP-based method.

## Hybrid Temporal Action Logic Theories

Description logics form a basis of the Web Ontology Language (OWL). We consider how to integrate TAL with description logic based ontologies. Since both of them are based on classical logic, the integration is straightforward.

**Definition 1** *Let $\Sigma_\Gamma$ and $\Sigma_\mathcal{T}$ be signatures which agree on common constants. A hybrid action theory $(\Gamma, \mathcal{T})$, where $\Gamma$ is a TAL domain description of signature $\Sigma_\Gamma$ and $\mathcal{T}$ is a description logic knowledge base of signature $\Sigma_\mathcal{T}$, is the conjunction $\Gamma \wedge FO(\mathcal{T})$ of signature $\Sigma_\Gamma \cup \Sigma_\mathcal{T}$, where $FO(\mathcal{T})$ is the first-order logic representation of $\mathcal{T}$.*

Consider the example discussed in the introduction. Following is a partial TAL description of the domain in the surface language. *Admin*$(d)$ represents the action of drug $d$ being administered. For simplicity, the duration of the action

is assumed to be equal to the time taken for the effects of the drug to wear off. Fluent *Condition*$(c)$ is used to represent if the patient is currently suffering from disease/disorder $c$, and fluent *SideEffect*$(d)$ is used to represent if the drug $d$ has a side-effect on the patient. The predicates in all capitals (MAY_TREAT, MAY_PREVENT, and CONTRAINDICATION) are description logic predicates (properties in the ontologies), and are time-independent. MAY_TREAT$(d, c)$ represents that drug $d$ may treat $c$, MAY_PREVENT$(d, c)$ represents that drug $d$ may prevent $c$, and CONTRAINDICATION$(d, c)$ represents that $c$ is a contraindication for administering drug $d$. The fluents $Ab_1$ and $Ab_2$ are used to represent the abnormalities, and these are treated as durational fluents with default value *false*. The other fluents are persistent.

**acs1** $[t_1, t_2]$ *Admin*$(d) \rightarrow$
$\Big([t_1]$ *Condition*$(c) \wedge \neg$*Ab*$_1(d, c) \wedge$
(MAY_TREAT$(d, c) \vee$ MAY_PREVENT$(d, c)) \wedge$
$\neg\exists c_1, t\big(t_1 \leq t \leq t_2 \wedge [t]$ *Condition*$(c_1) \wedge$
CONTRAINDICATION$(d, c_1)\big) \rightarrow I((t_1, t_2]\ \neg$*Condition*$(c))\Big)$.

**acs2** $[t_1, t_2]$ *Admin*$(d) \rightarrow$
$\forall t\big(t_1 \leq t \leq t_2 \wedge$
$[t]\exists c($*Condition*$(c) \wedge$ CONTRAINDICATION$(d, c))$
$\rightarrow I([t, t_2]$ *SideEffect*$(d))\big)$.

**dep1** $\forall t([t]$ *Condition*(*Gastritis*) $\wedge \neg$*Ab*$_2$(*Gastritis*) $\rightarrow$
$R([t + 1]$ *Condition*(*GastrointestinalBleeding*))).

The first statement represents that if no contraindication to the drug $d$ holds in the interval $[t_1, t_2]$, then administering $d$, which may treat or may prevent $c$, usually has a positive effect on the patient who is suffering from $c$. The second statement represents that if a contraindication to drug $d$ holds in the interval $[t_1, t_2]$, then administering $d$ in that interval causes a side-effect. Finally, the dependency constraint represents that gastritis usually causes gastroinstestinal bleeding. This example shows the usefulness of the durative actions and dependency constraints available in TAL.

Given that *Condition*(*Gastritis*) and *Condition*(*Pain*) hold at timepoint 0, and the ontology being queried entails the facts MAY_TREAT(*Aspirin*, *Pain*) and CONTRAINDICATION(*Aspirin*, *GastrointestinalBleeding*), one can verify the following from the above description:

- *Condition*(*GastrointestinalBleeding*) is true for all timepoints $\tau \geq 1$;
- if *Admin*(*Aspirin*) occurs in the interval $[0, 2]$, then *SideEffect*(*Aspirin*) is true for all timepoints $\tau \geq 1$.

Thus, the system will conclude that Aspirin is not recommended.

Now, consider an initial state in which only *Condition*(*Pain*) is true and the rest of the fluents are false. Further, assume that neither *Condition*(*GastrointestinalBleeding*) nor any other contraindication to administering aspirin is caused in some interval $[\tau_1, \tau_2]$. If *Admin*(*Aspirin*) occurs in the interval $[\tau_1, \tau_2]$, then *Condition*(*Pain*) is false for all $\tau > \tau_1$ and no side-effect will be caused. So, the system will be able to conclude that Aspirin can be recommended, unless a contraindication to it may be caused in the interval that it is to be administered.

## Using DLVHEX to Compute Hybrid Theories

We use the answer set solver DLVHEX for reasoning about hybrid theories as it allows for integrating with external ontologies. In our experiments, we use two ontologies represented in OWL: the NDF ontology (NDF-RT2 public edition), which contains knowledge about drugs (i.e., what they may treat, prevent, the contraindications for the drugs, etc.), and the human disease ontology (alpha version), which contains knowledge about human diseases (such as definitions of the diseases, synonyms of the diseases, etc.). The NDF ontology has over 40,000 classes and the human disease ontology has over 8000 classes.

We encode the TAL description in F2LP and use the rdf plugin for DLVHEX to generate the relevant instances of the Description Logic atoms MAY_TREAT$(d, c)$, MAY_PREVENT$(d, c)$, and CONTRAINDICATION$(d, c)$. Since F2LP only outputs programs in the language of GRINGO and LPARSE, we implemented a python program `f2lpdlv.py` to turn the output of F2LP into the language of DLV[17]. Also, since DLVHEX (v.1.7.2) does not allow functions, we flatten atoms (e.g., holds(T, condition(C), V) is turned into holds(T, condition, C, V)).

For the above example, we add the following observations and query:

```
%initial state
gastritisId(X) -> holds(0,condition,X,true).
painId(X) -> holds(0,condition,X,true).
giBleedingId(X) -> holds(0,condition,X,false).
drug(D) -> holds(0,sideEffect,D,false).

% query: can aspirin be given to the patient?
aspirinId(X) -> occurs(0,2,admin,X).
painId(C) & aspirinId(X) ->
-(holds(2,condition,C,false) &
  -?[T]: (T >=0 & T <=2 &
          holds(T,sideEffect,X,true))).
```

The predicates ending with Id represent the class names (in the NDF) of the corresponding drug or condition. For instance, the extent of painId contains just one element and that element is the class name of the class with label *Pain*. The first formula following the initial state description represents that Aspirin is administered in the interval $[0, 2]$, and the second formula is the negation of the query we want to check. The query here is to check if administering Aspirin treats the pain such that no side-effect is caused in the interval $[0, 2]$.

Following is a part of an answer set produced when the maximum timepoint is set to 2 (the existence of an answer set indicates that the query is not entailed):

```
{holds(0,condition,"<NDF#N0000001314>",true),
 holds(0,condition,"<NDF#N0000002278>",true),
 holds(1,condition,"<NDF#N0000001482>",true),
 holds(1,condition,"<NDF#N0000001314>",true),
 holds(1,condition,"<NDF#N0000002278>",true),
```

[17] http://www.dlvsystem.com/

```
holds(1,sideEffect,"<NDF#N0000145918>",true),
holds(2,condition,"<NDF#N0000001482>",true),
holds(2,condition,"<NDF#N0000001314>",true),
holds(2,condition,"<NDF#N0000002278>",true),
holds(2,sideEffect,"<NDF#N0000145918>",true)}
```

Here, N0000001314, N0000002278, N0000001482, and N0000145918 are the class names corresponding to *Gastritis*, *Pain*, *Gastroinstestinal Bleeding*, and *Aspirin* respectively. Also, NDF is an abbreviation for the namespace of the ontology. This output indicates that a side-effect is caused at timepoint 1 and that administering aspirin in the interval $[0, 2]$ does not relieve pain. Running the same query with the drug *acetaminophen* instead of aspirin produces no answer set, which indicates that acetaminophen can be recommended.

We also tested some other examples, details of which are available on the F2LP webpage.

## Conclusion

Our reformulation of TAL in ASP is based on the same idea as the reformulations of the event calculus and the situation calculus in ASP (Kim, Lee, & Palla 2009; Lee & Palla 2010; 2012). On the other hand, it is distinct from the earlier work as it shows that functional fluents in TAL can be represented by constraint variables in constraint answer set solving. Also, it shows that an answer set solver like DLVHEX can be used for computing the integration of TAL with ontologies, thereby allowing expressive reasoning about dynamic systems while also being able to query ontologies. We expect that the techniques developed in this paper can be applied to other action calculi as well.

## Acknowledgements

We are grateful to Jonas Kvarnström for his help on using VITAL and TALPlanner, and for answering our related questions. We are also grateful to the anonymous referees for their useful comments. This research was partially supported by the National Science Foundation under Grant IIS-0916116 and by Siemens Corporate Research. The work was done when the second author was at Arizona State University.

## References

Akman, V.; Erdoğan, S.; Lee, J.; Lifschitz, V.; and Turner, H. 2004. Representing the Zoo World and the Traffic World in the language of the Causal Calculator. *Artificial Intelligence* 153(1–2):105–140.

Doherty, P., and Kvarnström, J. 2008. Temporal action logics. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier Science. 709–757.

Doherty, P.; Gustafsson, J.; Karlsson, L.; and Kvarnström, J. 1998. TAL: Temporal action logics language specification and tutorial[18]. *Linköping Electronic Articles in Computer and Information Science ISSN 1401-9841* 3(015).

Doherty, P.; Kvarnström, J.; and Heintz, F. 2009. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Autonomous Agents and Multi-Agent Systems* 19(3):332–377.

Ferraris, P.; Lee, J.; Lifschitz, V.; and Palla, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 797–803.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

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

Gelfond, M., and Lifschitz, V. 1998. Action languages[19]. *Electronic Transactions on Artificial Intelligence* 3:195–210.

Kim, T.-W.; Lee, J.; and Palla, R. 2009. Circumscriptive event calculus as answer set programming. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 823–829.

Kvarnström, J. 2005. TALplanner and other extensions to temporal action logic. *PhD thesis, Linköping University, Linköping Studies in Science and Technology, Dissertation No. 937*.

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

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

Lee, J., and Palla, R. 2012. 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)* To appear.

Lifschitz, V. 1994. Circumscription. In Gabbay, D.; Hogger, C.; and Robinson, J., eds., *Handbook of Logic in AI and Logic Programming*, volume 3. Oxford University Press. 298–352.

Magnusson, M., and Doherty, P. 2008. Temporal action logic for question answering in an adventure game. In *In Proc. 1st Artificial Game Intelligence (AGI) Conference*, 236–247.

McCarthy, J. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence* 13:27–39,171–172.

McCarthy, J. 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 26(3):89–116.

Rosse, C., and Mejino, J. L. V. 2003. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics* 36(6):478–500.

Sandewall, E. 1994. *Features and Fluents*, volume 1. Oxford University Press.

[18] http://www.ep.liu.se/ea/cis/1998/015/

[19] http://www.ep.liu.se/ea/cis/1998/016/