Dynamic and Temporal Answer Set Programming on Linear (Finite) Traces

Pedro Cabalar and Torsten Schaub

University of Corunna, Spain

University of Potsdam, Germany
Objective

Extend Answer Set Programming (ASP) with means for representing and reasoning about dynamic knowledge.
Introduction

- **Objective**
  
  Extend *Answer Set Programming* (ASP) with means for representing and reasoning about dynamic knowledge

- **Approach**
  
  Extend the base logic of ASP, namely the *logic of Here-and-There* (HT), with language elements from
  - Linear Temporal Logic (LTL)
  - Linear Dynamic Logic (LDL)
  
  over a common semantic structure, namely, (finite) HT traces
Objective

Extend Answer Set Programming (ASP) with means for representing and reasoning about dynamic knowledge

Approach

Extend the base logic of ASP, namely the logic of Here-and-There (HT), with language elements from

- Linear Temporal Logic (LTL)
- Linear Dynamic Logic (LDL)

over a common semantic structure, namely, (finite) HT traces

Origin

Temporal logic of Here-and-There (Cabalar and Pérez, 2007) over infinite traces
The logic of Here-and-There

- **Origin**

  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective
The logic of Here-and-There

- **Origin**

  Three valued logic due to (Heyting, 1930; Gödel, 1932)

  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective, and
    negation is defined in terms of implication: \(\neg \varphi = \varphi \rightarrow \bot\)
The logic of Here-and-There

- **Origin**

  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective
The logic of Here-and-There

■ **Origin**

Three valued logic due to (Heyting, 1930; Gödel, 1932)

■ HT is based on Kripke semantics for intuitionistic logic

■ An HT model is a pair \((H, T)\) such that \(H \subseteq T\)

■ Implication is a genuine connective

■ **Discovery** (Pearce, 1996)

Minimal HT models correspond to answer sets
The logic of Here-and-There

- **Origin**
  
  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective

- **Discovery** (Pearce, 1996)
  
  Minimal HT models correspond to answer sets, more precisely, an answer set \(T\) of \(\phi\) is
  
  - a total HT model \((T, T)\) of \(\phi\) and
  - there is no \(H \subset T\) such that \((H, T)\) is an HT model of \(\phi\)
The logic of Here-and-There

- **Origin**
  
  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective

- **Discovery** (Pearce, 1996)

  Minimal HT models correspond to answer sets, more precisely, an answer set \(T\) of \(\phi\) is
  
  - a total HT model \((T, T)\) of \(\phi\) and
  - there is no \(H \subset T\) such that \((H, T)\) is an HT model of \(\phi\)

  Such models are are called **equilibrium models**
The logic of Here-and-There

- **Origin**

  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective

- **Discovery** (Pearce, 1996) — Equilibrium logic

  Minimal HT models correspond to answer sets, more precisely, an answer set \(T\) of \(\phi\) is
  - a total HT model \((T, T)\) of \(\phi\) and
  - there is no \(H \subset T\) such that \((H, T)\) is an HT model of \(\phi\)

  Such models are are called **equilibrium models**
The logic of Here-and-There

- **Origin** (monotonic)
  
  Three valued logic due to (Heyting, 1930; Gödel, 1932)
  
  - HT is based on Kripke semantics for intuitionistic logic
  - An HT model is a pair \((H, T)\) such that \(H \subseteq T\)
  - Implication is a genuine connective

- **Discovery** (Pearce, 1996) — Equilibrium logic (non-monotonic)
  
  Minimal HT models correspond to answer sets, more precisely, an answer set \(T\) of \(\phi\) is
  
  - a total HT model \((T, T)\) of \(\phi\) and
  - there is no \(H \subset T\) such that \((H, T)\) is an HT model of \(\phi\)

- Such models are called equilibrium models
Dynamic and Temporal Logic of Here-and-There
Dynamic and Temporal Logic of Here-and-There

- Structure: An HT trace is a sequence $(H_i, T_i)_{i=0}^\lambda$ of HT models.
Dynamic and Temporal Logic of Here-and-There

- **Structure** An HT trace is a sequence \((H_i, T_i)_{i=0}^{\lambda}\) of HT models
- **Satisfaction** \((H, T) = (H_i, T_i)_{i=0}^{\lambda}\)
Dynamic and Temporal Logic of Here-and-There

- **Structure** An HT trace is a sequence \((H_i, T_i)_{i=0}^{\lambda}\) of HT models.

- **Satisfaction** \((H, T) = (H_i, T_i)_{i=0}^{\lambda}\)
  
  - **Something Boolean** \((H, T), k \models \varphi \rightarrow \psi\) if
    
    \((H', T), k \not\models \varphi\) or \((H', T), k \models \psi\), for all \(H' \in \{H, T\}\)
Dynamic and Temporal Logic of Here-and-There

- **Structure** An HT trace is a sequence \((H_i, T_i)_{i=0}^{\lambda}\) of HT models

- **Satisfaction** \((H, T) = (H_i, T_i)_{i=0}^{\lambda}\)
  - **Something Boolean** \((H, T), k \models \varphi \rightarrow \psi\) if
    \((H', T), k \not\models \varphi\) or \((H', T), k \models \psi\), for all \(H' \in \{H, T\}\)
  - **Something Temporal**
    \((H, T), k \models \Box \varphi\) if \((H, T), i \models \varphi\) for any \(i = k..\lambda\)
    \((H, T), k \models \Diamond \varphi\) if \((H, T), i \models \varphi\) for some \(i = k..\lambda\)
Dynamic and Temporal Logic of Here-and-There

- **Structure** An HT trace is a sequence \((H_i, T_i)_{i=0}^\lambda\) of HT models

- **Satisfaction** \((H, T) = (H_i, T_i)_{i=0}^\lambda\)
  - Something Boolean \((H, T), k \models \varphi \rightarrow \psi\) if
    \((H', T), k \not\models \varphi\) or \((H', T), k \models \psi\), for all \(H' \in \{H, T\}\)
  - Something Temporal
    \((H, T), k \models \Box \varphi\) if \((H, T), i \models \varphi\) for any \(i = k..\lambda\)
    \((H, T), k \models \Diamond \varphi\) if \((H, T), i \models \varphi\) for some \(i = k..\lambda\)
  - Something Dynamic \((H, T), k \models [\rho] \varphi\) if
    \((H', T), i \models \varphi\) for all \(i = 0..\lambda\) with \((k, i) \in \parallel \rho \parallel^{(H', T)}\) and \(H' \in \{H, T\}\)
telingo

- telingo
  - extends the full modeling language of **clingo** with (past and future) temporal operators
  - relies on finite trace
  - implements an incremental translation

Example: "A robot cannot lift a box unless its capacity exceeds the box's weight plus that of all held objects"

:- lift(R,B), robot(R), capacity(R,C), # sum { W : box(B,W); V,O : ' holding(R,O,V) } > C.
telingo

- telingo — https://github.com/potassco/telingo
- extends the full modeling language of clingo with (past and future) temporal operators
- relies on finite trace
- implements an incremental translation

Example: "A robot cannot lift a box unless its capacity exceeds the box's weight plus that of all held objects" :- lift (R,B), robot (R), capacity (R,C), # sum { W : box (B,W); V,O : ' holding (R,O,V) } > C.
telingo

- telingo — https://github.com/potassco/telingo
  - extends the full modeling language of clingo
    with (past and future) temporal operators
  - relies on finite trace
  - implements an incremental translation

- Primes allow for expressing (iterated) next and previous operators
  - $p(a)$ and $q(b)$ can be expressed by $p'(a)$ and $q'(b)$

Example

"A robot cannot lift a box unless its capacity exceeds the box's weight plus that of all held objects"}

:- lift(R,B), robot(R), capacity(R,C),
     # sum { W : box(B,W); V,O : holding(R,O,V) } > C.
**telingo**

- telingo — https://github.com/potassco/telingo

  - extends the full modeling language of clingo
    with (past and future) temporal operators
  - relies on finite trace
  - implements an incremental translation

- Primes allow for expressing (iterated) next and previous operators
  - \(p(a)\) and \(q(b)\) can be expressed by \('p(a)\) and \(q'(b)\)

- Example  "A robot cannot lift a box unless its capacity exceeds the box's weight plus that of all held objects"

  ```prolog
  :- lift(R,B), robot(R), capacity(R,C),
  #sum {   W : box(B,W); 
  V,O : 'holding(R,O,V) } > C.
  ```
Wolf, sheep, and cabbage

```prolog
#program always.
item(w;s;c).
opp(l,r). opp(r,l).
eats(w,s). eats(s,c).

#program initial.
at(b,l).
at(X,l) :- item(X).  \% everything at the left bank

#program dynamic.
at(X,A) :- 'at(X,B), m(X), opp(A,B).  \% effect axiom for moving item X
at(b,A) :- 'at(b,B), opp(A,B).  \% boat is always moving
at(X,A) :- 'at(X,A), not at(X,B), opp(A,B).  \% inertia
0 { m(X) : item(X) } 1.  \% choose moving at most one item

#program always.
:- m(X), 'at(b,A), 'at(X,B), opp(A,B).  \% we cannot move item X if at the opposite bank
:- eats(X,Y), at(X,A), at(Y,A), opp(A,B), at(b,B).  \% we cannot leave them alone

#program final.
:- at(X,l).

#show m/1.
```

Torsten Schaub (KRR@UP)  Dynamic and Temporal ASP  6 / 8
telingo’s solution

$ telingo version 1.0
Reading from wolf.tel
Solving...
Solving...
Solving...
Solving...
Solving...
Solving...
Solving...
Solving...
Answer: 1
  State 0:
    State 1: m(s)
    State 2:
    State 3: m(w)
    State 4: m(s)
    State 5: m(c)
    State 6:
    State 7: m(s)
Answer: 2
  State 0:
    State 1: m(s)
    State 2:
    State 3: m(c)
    State 4: m(s)
    State 5: m(w)
    State 6:
    State 7: m(s)
SATISFIABLE

Models : 2
Calls : 8
Time : 0.156s (Solving: 0.00s)
CPU Time : 0.028s
Summary

- **ASP + Temporal and Dynamic Logic**
  
  via extending HT over the common semantic structure of HT traces
Summary

- **ASP + Temporal and Dynamic Logic**
  
  via extending HT over the common semantic structure of HT traces

- Interested? JANCL’13, ICLP’18, KR’18
Summary

- **ASP + Temporal and Dynamic Logic**
  via extending HT over the common semantic structure of HT traces

- Interested? JANCL’13, ICLP’18, KR’18

- Playful? [https://github.com/potassco/telingo](https://github.com/potassco/telingo)
Summary

- **ASP + Temporal and Dynamic Logic**
  
  via extending HT over the common semantic structure of HT traces

- Interested? JANCL'13, ICLP'18, KR'18

- Playful?[^1] [https://github.com/potassco/telingo](https://github.com/potassco/telingo)

[^1]: Classical logic is obtained in ASP by adding choices; eg., '{a}' stands for 'a ∨ ¬a.'