Online appendix for the paper

Reasoning about Actions with Temporal Answer Sets

published in Theory and Practice of Logic Programming

LAURA GIORDANO
Dipartimento di Informatica, Università del Piemonte Orientale, Italy
laura@mfn.unipmn.it

ALBERTO MARTELLI
Dipartimento di Informatica, Università di Torino, Italy
mrt@di.unito.it

DANIELE THESEIDER DUPRÉ
Dipartimento di Informatica, Università del Piemonte Orientale, Italy
dtd@mfn.unipmn.it

submitted 15 November 2010; revised 16 April 2011; accepted 31 August 2011

Appendix A

We prove Theorem 1 and Theorem 2. Let $(\Pi, \mathcal{C})$ be a well-defined domain description over Σ. We show that there is a one to one correspondence between the temporal answer sets of Π and the answer sets of the translation $tr(\Pi)$.

Theorem 1

(1) Given a temporal answer set (σ, S) of Π such that σ can be finitely represented as a finite path with a k-loop, there is a consistent answer set R of $tr(\Pi)$ such that R and S correspond to the same temporal model.
(2) Given a consistent answer set R of $tr(\Pi)$, there is a temporal answer set (σ, S) of Π (that can be finitely represented as a finite path with a back loop) such that R and S correspond to the same temporal model.

Proof
Let us prove item (1). Let (σ, S) be a temporal answer set of Π such that σ can be finitely represented as a finite path with a back loop, i.e.,

$$\sigma = a_1 a_2 \dots a_j a_{j+1} \dots a_{k+1} a_{j+1} \dots a_{k+1} \dots$$

We construct an answer set R of $tr(\Pi)$ as follows. R contains the following literals:

$state(0), \ldots, state(k)$
$next(0,1), next(1,2), \ldots, next(k-1,k), next(k,j)$, $\neg next(k,s)$, for all $s \neq j$
$occurs(a_1, 0), occurs(a_2, 1), ..., occurs(a_{j+1}, j), ..., occurs(a_{k+1}, k)$
$\neg occurs(a,s)$, for all other ground instances of predicate *occurs*,
$eq\_last(j)$
for all $i = 0, \ldots, k$, for all fluent names $f \in \mathcal{P}$:

$$(\neg) holds(f, i) \in R \text{ if and only if } [a_1; \ldots; a_i](\neg) f \in S$$

From the consistency of S, it is easy to see that R is a consistent set of literals. To show that R is an answer set of $tr(\Pi)$, we show that: (i) R is closed under $tr(\Pi)^R$; (ii) R is minimal (in the sense of set inclusion) among the consistent sets of literals closed under $tr(\Pi)^R$.

(i) For all the rules r in $tr(\Pi)^R$, we have to prove that if the literals in the body of r belong to R, then the head of r belongs to R. Let us consider the case when the rule r in $tr(\Pi)^R$ is obtained by translating an action law in Π, of the form:

$$\Box([a](\neg) f \leftarrow t_1, \ldots, t_m, not\ t_{m+1}, \ldots, not\ t_n)$$

(the other cases are similar). In this case, $tr(\Pi)$ contains the translation of the action law above:

$$(\neg) holds(f, S') \leftarrow state(S), S' = S + 1, occurs(a, S), h_1 \ldots h_m, not\ h_{m+1} \ldots not\ h_n$$

where $h_i = (\neg) holds(f_i, S')$ if $t_i = [a](\neg) f_i$ or $h_i = (\neg) holds(f_i, S)$ if $t_i = (\neg) f_i$.

Let us consider the ground instantiation of the rule above from which r is obtained. Suppose S is instantiated with some $s \in \{1, \ldots, k\}$. It must be the case that $a = a_{s+1}$, as $occurs(a_{s+1}, s) \in R$ and no other action occurs in state s according to R.

If the rule r:

$$(\neg) holds(f, s+1) \leftarrow state(s), occurs(a_{s+1}, s), h'_1 \ldots h'_m \tag{A1}$$

belongs to the reduct $tr(\Pi)^R$ (where each h'_t is the ground instantiation of h_t with $S = s$), then $h'_{m+1} \notin R, \ \ldots, \ h'_n \notin R$.

We have to show that, if the body of (A1) belongs to R then its head also belongs to R.

Assume $h'_1, \ldots, h'_m$ belong to R. For each $i = 1 \ldots, m$, either $h'_i = (\neg) holds(f_i, s)$ (if $t_i = (\neg) f_i$) or $h'_i = (\neg) holds(f_i, s+1)$ (if $t_i = [a](\neg) f_i$). If $h'_i = (\neg) holds(f_i, s)$, by construction of R, $[a_1; \ldots; a_s](\neg) f_i \in S$, i.e., $(\sigma, S), a_1, \ldots, a_s \models (\neg) f_i$, and hence, $(\sigma, S), a_1, \ldots, a_s \models t_i$. If $h'_i = (\neg) holds(f_i, s+1)$, by construction of R, $[a_1; \ldots; a_s; a_{s+1}](\neg) f_i \in S$, i.e., $(\sigma, S), a_1, \ldots, a_s, a_{s+1} \models (\neg) f_i$, hence, $[a_1; \ldots; a_s; a_{s+1}](\neg) f_i \in S$, and then $(\sigma, S), a_1, \ldots, a_s \models [a_{s+1}](\neg) f_i$. Thus, $(\sigma, S), a_1, \ldots, a_s \models t_i$. So the positive literals in the temporal action law are satisfied.

To show that the negated literals $t_{m+1}, \ldots, t_n$ in the body of the temporal clause are not satisfied in (σ, S) at $a_1, \ldots, a_s$, consider the fact that $h'_{m+1} \notin R, \ \ldots, \ h'_n \notin R$. Again, for each $i = m+1, \ldots, k$, either $h'_i = (\neg) holds(f_i, s))$ or $h'_i = (\neg) holds(f_i, s+1)$.

If $h'_i = (\neg) holds(f_i, s) \notin R$, by construction of R, $[a_1; \ldots; a_s](\neg) f_i \notin S$, i.e.,

$(\sigma, S), a_1, \ldots, a_s \not\models (\neg)f_i$, and hence, $(\sigma, S), a_1, \ldots, a_s \not\models t_i$. If $h'_i = (\neg)holds(f_i, s+1) \notin R$, by construction of R, $[a_1; \ldots; a_s; a_{s+1}](\neg)f_i \notin S$, hence, $(\sigma, S), a_1, \ldots, a_s \not\models [a_{s+1}](\neg)f_i$. Thus, $(\sigma, S), a_1, \ldots, a_s \not\models t_i$.

We have shown that the body of the temporal rule

$$\Box([a](\neg)f \leftarrow t_1, \ldots, t_m, not\ t_{m+1}, \ldots, not\ t_n)$$

is true in (σ, S) at $a_1, \ldots, a_s$, i.e.,

$$(\sigma, S), a_1, \ldots, a_s \models t_1, \ldots, t_m, not\ t_{m+1}, \ldots, not\ t_n$$

As the temporal rule belongs to Π and is satisfied in (σ, S), we can conclude that its head is also satisfied in $a_1, \ldots, a_s$, i.e., $(\sigma, S), a_1, \ldots, a_s \models [a](\neg)f$, namely, $[a_1; \ldots; a_s; a](\neg)f \in S$. As we observed above, $a = a_{s+1}$, hence, $[a_1; \ldots; a_s; a_{s+1}](\neg)f \in S$ and, by construction of R, $(\neg)holds(f, s+1) \in R$.

To prove (ii), we have to show that R is minimal (in the sense of set inclusion) among the consistent sets of literals closed under $tr(\Pi)^R$. Suppose R is not minimal, and there is a consistent set of literals R' which is closed under $tr(\Pi)^R$ and such that $R' \subset R$.

Suppose there is a literal $A \in R$ such that $A \notin R'$. For the auxiliary predicates *occurs*, *next*, etc., it is easy to see that this cannot be the case. Let us consider the case $A = (\neg)holds(f, i)$ and suppose that $(\neg)holds(f, i) \in R$ and $(\neg)holds(f, i) \notin R'$.

We show that we can construct from R' an $S' \subset S$ such that (σ, S') satisfies the rules in $\Pi^{(\sigma,S)}$. We define S' as follows:

$$[a_1; \ldots; a_h](\neg)f \in S' \text{ if and only if } (\neg)holds(f, h) \in R'$$

It can be shown that (σ, S') satisfies the rules in $\Pi^{(\sigma,S)}$. In fact, for each rule r in $\Pi^{(\sigma,S)}$ whose body is satisfied in (σ, S'), there is a rule r' in $tr(\Pi)^R$, whose body is true in R'. As R' is closed under $tr(\Pi)^R$, the head of r' must be true in R'. By construction of S', the head of r is satisfied in (σ, S').

As $S' \subset S$ and (σ, S') satisfies the rules in $\Pi^{(\sigma,S)}$, S is not minimal among the interpretations S'' such that (σ, S'') satisfies the rules in $\Pi^{(\sigma,S)}$. This contradicts the hypothesis that (σ, S) is a temporal answer set of Π.

As the domain description is well-defined, (σ, S) has to be a total temporal answer set. Hence, for each state $i = 1, \ldots, k$, either $holds(p, i) \in R$ or $\neg holds(p, i) \in R$. It is easy to see that R and (σ, S) correspond to the same temporal model, as M_S and M_R are defined over the same sequence σ and, for each finite prefix τ of σ, they give the same evaluation to atomic propositions in τ.

Let us prove item (2).

Let R be an answer set of $tr(\Pi)$. We define a temporal answer set (σ, S) of Π as follows.

Given the definition of the predicates *next* and *occurs* in $tr(\Pi)$, R must contain, for some k and j, and for some $a_1, \ldots, a_{k+1}$, the literals:
$next(0, 1), next(1, 2), \ldots, next(k-1, k), next(k, j)$,
$occurs(a_1, 0), occurs(a_2, 1), ..., occurs(a_{j+1}, j), ..., occurs(a_{k+1}, k)$,

$eq\_last(j)$.
We define σ as:

$$\sigma = a_1 a_2 \dots a_j a_{j+1} \dots a_{k+1} a_{j+1} \dots a_{k+1} \dots$$

We determine the temporal literals that belong to S as follows: for all $i = 0, \dots, k$ for all fluent names $f \in \mathcal{P}$:

$$[a_1; \dots; a_i](\neg)f \in S \text{ if and only if } (\neg)holds(f, i) \in R$$

From the consistency of R, it is easy to see that S is a consistent set of temporal literals. To show that S is a temporal answer set of Π, we show that:
(i) (σ, S) satisfies all the rules in $\Pi^{(\sigma,S)}$;
(ii) S is minimal (in the sense of set inclusion) among the S' such that (σ, S') is a partial interpretation satisfying the rules in $\Pi^{(\sigma,S)}$.

(i) Let us prove that (σ, S) satisfies all the rules in $\Pi^{(\sigma,S)}$. Let

$$[a_1, \dots, a_s](H \leftarrow t_1, \dots, t_m)$$

be a rule in $\Pi^{(\sigma,S)}$, where $a_1, \dots a_s \in prf(\sigma)$. Then there must be a law in Π of the form:

$$\Box(H \leftarrow t_1, \dots, t_m, not\ t_{m+1}, \dots, not\ t_n)$$

such that $(\sigma, S), a_1 \dots a_s \not\models t_i$, for $i = m+1, \dots, n$.

Let us consider the case where such a law is a dynamic causal law, (the other cases are similar). In this case, $H = \bigcirc(\neg)f$ and the law has the form:

$$\Box(\bigcirc(\neg)f \leftarrow t_1, \dots, t_m, not\ t_{m+1}, \dots, not\ t_n)$$

where, for all $i = 1, \dots, n$, $t_i = (\neg)f_i$ or $t_i = \bigcirc(\neg)f_i$.

Then, $tr(\Pi)$ contains its translation:

$$(\neg)holds(f, S') \leftarrow state(S), S' = S + 1, h_1 \dots h_m, not\ h_{m+1} \dots not\ h_n$$

where $h_i = (\neg)holds(f_i, S')$ (if $t_i = \bigcirc(\neg)f_i$) or $h_i = (\neg)holds(f_i, S)$ (if $t_i = (\neg)f_i$).

Let us consider the ground instantiation of the rule above with $S = s$, for some $s \in \{1, \dots, k\}$.

$$(\neg)holds(f, s+1) \leftarrow state(s), h'_1 \dots h'_m, not\ h'_{m+1} \dots not\ h'_n$$

where $h'_i = (\neg)holds(f_i, s+1)$ (if $t_i = \bigcirc(\neg)f_i$) or $h'_i = (\neg)holds(f_i, s)$ (if $t_i = (\neg)f_i$).

The rule

$$(\neg)holds(f, s+1) \leftarrow state(s), h'_1 \dots h'_m \tag{A2}$$

must belong to the reduct $tr(\Pi)^R$. In fact, we can prove that $h'_{m+1} \notin R$, $\dots$, $h'_n \notin R$. Let $t_i = (\neg)f_i$ and $h'_i = (\neg)holds(f_i, s)$. From the hypothesis, we know that $(\sigma, S), a_1 \dots a_s \not\models t_i$, i.e., $(\sigma, S), a_1 \dots a_s \not\models (\neg)f_i$, i.e., $[a_1; \dots; a_s](\neg)f_i \notin S$. As, by construction of (σ, S), $(\neg)holds(f_i, s) \in R$ iff $[a_1; \dots; a_s](\neg)f_i \in S$, we conclude $(\neg)holds(f_i, s) \notin R$. Let $t_i = \bigcirc(\neg)f_i$ and $h'_i = (\neg)holds(f_i, s+1)$. From the hypothesis, we know that $(\sigma, S), a_1 \dots a_s \not\models t_i$, i.e., $(\sigma, S), a_1 \dots a_s \not\models \bigcirc(\neg)f_i$, i.e.,

$[a_1;\ldots;a_s;a_{s+1}](\neg)f_i \notin S$. As, by construction of (σ,S), $(\neg)holds(f_i,s+1) \in R$ iff $[a_1;\ldots;a_{s+1}](\neg)f_i \in S$, we conclude $(\neg)holds(f_i,s+1) \notin R$, that is $h'_i \notin R$.

To show that the law

$$[a_1,\ldots,a_s](H \leftarrow t_1,\ldots,t_m)$$

in $\Pi^{(\sigma,S)}$ is satisfied in (σ,S), let us assume that its body is satisfied in (σ,S), that is, $(\sigma,S),a_1\ldots a_s \models t_1,\ldots,t_m$, i.e., $(\sigma,S),a_1\ldots a_s \models t_i$, for all $i=1,\ldots,m$. By the same pattern of reasoning as above, we can show that $h'_i \in R$, for all $i=1,\ldots,m$. As rule (A2) is in $tr(\Pi)^R$, its body is true in R, and R is closed under $tr(\Pi)^R$, then the head of (A2), $(\neg)holds(f,s+1)$, belongs to R. Hence, by construction of (σ,S), $[a_1;\ldots;a_s] \bigcirc (\neg)f \in S$, that is $(\sigma,S),a_1\ldots a_s \models H$, namely, the head of the rule

$$[a_1,\ldots,a_s](H \leftarrow t_1,\ldots,t_m)$$

is satisfied in (σ,S).

(ii) S is minimal (in the sense of set inclusion) among the S' such that (σ,S') is a partial interpretation satisfying the rules in $\Pi^{(\sigma,S)}$.

Assume by contradiction that S is not minimal. Then, there is a partial interpretation (σ,S'), with $S' \subset S$, satisfying the rules in $\Pi^{(\sigma,S)}$.

We show that we can construct an $R' \subset R$ such that R' is closed under $tr(\Pi)^R$. We define R' as R, but for the predicate *holds*, for which we have:

$$(\neg)holds(f,h) \in R' \text{ if and only if } [a_1;\ldots;a_h](\neg)f \in S'$$

It can be shown that R' is closed under $tr(\Pi)^R$. In fact, for each rule r in $tr(\Pi)^R$ whose body is true in R', there is a rule r' in $\Pi^{(\sigma,S)}$, whose body is satisfied in (σ,S'). As (σ,S') satisfies all the rules in $\Pi^{(\sigma,S)}$, the head of r' must be satisfied in (σ,S'). By construction of R', the head of r belongs to R'.

As $R' \subset R$ and R' is closed under $tr(\Pi)^R$, R is not minimal among the consistent sets of literals closed under $tr(\Pi)^R$. This contradicts the hypothesis that R is an answer set of $tr(\Pi)$.

To prove that R and (σ,S) correspond to the same temporal model we can use the same argument as for item (1). □

Theorem 2

Let Π be the set of laws of a well-defined domain description, R an answer set of $tr(\Pi)$ and α a DLTL formula. The temporal model $M_R=(\sigma,V)$ associated with R satisfies α if and only if there is an answer set R' of $tr'(\Pi)$ such that $R \subset R'$ and $sat(t\_alpha,0) \in R'$ (where t_alpha is the term representing the formula α and *trans* and *final* encode the automata indexing the until formulas in α).

Proof

We first prove the "only if" direction of the theorem. We know by Theorem 1 that each answer set R of $tr(\Pi)$ corresponds to a temporal answer set of Π and, for each state $i=1,\ldots,k$, either $holds(p,i) \in R$ or $\neg holds(p,i) \in R$. Let us consider the temporal model $M_R=(\sigma_R,V_R)$ associated with R, as defined in section 7.

We extend R to define an answer set R' of $tr'(\Pi)$ as follows:

- all the literals in R belong to R';
- for all subformulas β of α, for all states $h \in \{0, \ldots, k\}$:

$$sat(t\_beta, h) \in R' \text{ if and only if } M_R, \tau_h \models \beta \tag{A3}$$

  where $\tau_h = a_1 \ldots a_h$ and t_beta is the term encoding the formula β.
- For each automaton $aut = (Q, \delta, Q_F)$ indexing an *until* formula in α:

$$final(aut, q) \in R' \text{ if and only if } q \in Q_f \tag{A4}$$

$$trans(aut, q_1, a, q_2) \in R' \text{ if and only if } q_2 \in \delta(q_1, a) \tag{A5}$$

We can show that R' is an answer set of $tr'(\Pi)$, i.e., (i) R' is closed under $tr'(\Pi)^{R'}$ (ii)R' is minimal among the consistent sets of literals closed under $tr'(\Pi)^{R'}$.

(i) holds trivially for all the rules in $tr(\Pi)$. It has to be proved for all the rules defining the predicate *sat*. We can procede by cases:

Let us consider the rule for fluents. Suppose R' satisfies the body of a ground instance of the rule:

$sat(F, S) : -fluent(F), holds(F, S).$

that is, $fluent(p) \in R'$ and $holds(p, h) \in R'$, for some fluent name p and some $h \in \{1, \ldots, k\}$. Then, $holds(p, h) \in R$, and thus $M_r, \tau_h \models p$. By construction of R', it must be: $sat(p, h) \in R'$.

Let us consider the first rule for until. Suppose R' satisfies the body of a ground instance of the rule:

$sat(until(Aut, Q, Alpha, Beta), S) : -final(Aut, Q), sat(Beta, S).$

that is, for some *aut* encoding a finite automaton $\mathcal{A} = (Q, \delta, Q_F)$, for some $q \in Q$, for some formula t_beta and state h, $final(aut, q) \in R'$ (i.e., $q \in Q_F$) and $sat(t\_beta, h) \in R'$. By construction of R', $M_R, \tau_h \models \beta$. As q is a final state of the finite automaton $\mathcal{A}$, it must be that $M_R, \tau_h \models \alpha \mathcal{U}^{\mathcal{A}(q)} \beta$. Hence, by construction,

$sat(until(aut, q, t\_alpha, t\_beta), h) \in R'$.

The other cases are similar.

(ii) We prove that R' is minimal among the consistent sets of literals closed under $tr'(\Pi)^{R'}$. Let us suppose that R' is not minimal and that there is an $R'' \subset R'$ such that R'' is closed with respect to $tr'(\Pi)^{R'}$. There must be a literal $l \in R' - R''$. l cannot be a literal in R, as R is an answer set of $tr(\Pi)$, and the definition of the predicates in R does not depend on the predicates *sat*, *trans* and $final$ introduced in $tr'(\Pi)$. Also, l cannot be a *trans* and $final$ literal, as these predicates are only defined by ground atomic formulas, which must be all in R''. Suppose there is $sat(t\_alpha, h) \in R'$ such that $sat(t\_alpha, h) \notin R''$.

Using the fact that R'' is closed with respect to $tr'(\Pi)^{R'}$, it can be proved that, for all the subformulas β of α, if $M_R, \tau_h \models \beta$ then $sat(t\_beta, h) \in R''$. The proof is by induction on the structure of β.

As $sat(t\_alpha, h) \in R'$, by construction of R' it must be that $M_R, \tau_h \models \alpha$. Then, by the previous property, $sat(t\_alpha, h) \in R''$. This contradicts the fact that $sat(t\_alpha, h) \notin R''$. Hence, R' is an answer set of $tr(\Pi)$.

To conclude the proof of the "only if" part, it is easy to see that, from (A3), if $M_R, \varepsilon \models \alpha$ then $sat(t\_alpha, 0) \in R'$, where ε represents the empty sequence of actions.

We have shown that, given an answer set R of $tr(\Pi)$ satisfying α we can construct an answer set R' of $tr'(\Pi)$ such that $sat(t\_alpha, 0) \in R'$. To prove the "if" direction of the theorem, let us assume that there is an answer set R'' of $tr'(\Pi)$ such that R'' extends R and $sat(t\_alpha, 0) \in R''$. We can show that R'' must coincide with R' built above. In fact, it can be easily proved that, for all subformulas β of α,

$$sat(t\_beta, h) \in R'' \text{ iff } sat(t\_beta, h) \in R'$$

The proof can be done by induction on the structure of β (observe that both R' and R'' extend R, which provides the evaluation of fluent formulas to be used by the sat predicate). As R'' coincides with R', if $sat(t\_alpha, 0) \in R''$ then by (A3), $M_R, \varepsilon \models \alpha$. $\square$

Appendix B

In this appendix we provide the encoding of BMC and Example 2 in DLV-Complex (https://www.mat.unical.it/dlv-complex).

```
state(0..#maxint).
laststate(N):- state(N), #maxint=N+1.

% general rules

occurs(A,S):- not ~occurs(A,S), action(A),state(S),laststate(L),S<=L.
~occurs(B,S):- occurs(A,S), action(A),state(S),action(B),A!=B.

next(S,SN):- state(S), laststate(LS), S<LS, SN=S+1.
-next(LS,S):- laststate(LS), next(LS,SS), state(S), state(SS), S!=SS.
next(LS,S):- laststate(LS), state(S), S<=LS, not -next(LS,S).
:- laststate(LS), next(LS,S), not eq_last(S).

diff_last(S):- state(S), S<#maxint, fluent(F),
               holds(F,S), -holds(F,#maxint).
diff_last(S):- state(S), S<#maxint, fluent(F),
               holds(F,#maxint), -holds(F,S).
eq_last(S):- state(S),  S<#maxint, not diff_last(S).

% The action theory makes use of the predicates:
% action(A), fluent(FL), holds(FL,State)

% evaluation of DLTL formulas
% makes use of predicate formula(F)

% true
sat(true,S):- state(S).

% fluents
sat(F,S):- fluent(F), state(S), holds(F,S).

% not
sat(neg(Alpha),S):- formula(neg(Alpha)), state(S), not sat(Alpha,S).

% or
sat(or(Alpha1,Alpha2),S):- formula(or(Alpha1,Alpha2)), state(S),
                           sat(Alpha1,S).
sat(or(Alpha1,Alpha2),S):- formula(or(Alpha1,Alpha2)), state(S),
                           sat(Alpha2,S).

% until
% An automaton is specified by the predicates
% trans(Automaton,Q1,Action,Q2)  and
% final(Automaton,Q)

sat(until(Aut,Q,Alpha,Beta),S):-
        formula(until(Aut,Q,Alpha,Beta)),
        final(Aut,Q),
        sat(Beta,S),
        state(S).
sat(until(Aut,Q,Alpha,Beta),S):-
        formula(until(Aut,Q,Alpha,Beta)),
        sat(Alpha,S),
        trans(Aut,Q,Act,Q1),
        action(Act),
        occurs(Act,S),
        next(S,S1),
        sat(until(Aut,Q1,Alpha,Beta),S1).


% derived operators and modalities
% ev(Alpha) means <>Alpha
% diamond(Az,Alpha) means <Az>Alpha
% box(Az,Alpha)  means  [Az]Alpha

sat(and(Alpha1,Alpha2),S):- formula(and(Alpha1,Alpha2)),
        state(S),
        sat(Alpha1,S), sat(Alpha2,S).

sat(impl(Alpha1,Alpha2),S):- formula(impl(Alpha1,Alpha2)),
        state(S),
        not sat(Alpha1,S).
sat(impl(Alpha1,Alpha2),S):- formula(impl(Alpha1,Alpha2)),
        state(S),
        sat(Alpha2,S).

sat(diamond(A,Alpha),S):- formula(diamond(A,Alpha)),
        action(A), state(S),
        occurs(A,S),
        next(S,SN),
        sat(Alpha,SN).

sat(ev(Alpha),S):- formula(ev(Alpha)),
        state(S),
        sat(Alpha,S).
sat(ev(Alpha),S):- formula(ev(Alpha)),
        state(S),
        next(S,SN),
        sat(ev(Alpha),SN).

sat(box(A,Alpha),S):- formula(box(A,Alpha)),
        action(A), state(S), action(B), formula(Alpha),
        occurs(B,S),
        A!=B.
sat(box(A,Alpha),S):- formula(box(A,Alpha)),
        state(S),
        occurs(A,S),
        next(S,SN),
        sat(Alpha,SN).

% the following rules define all subformulas of a given formula

formula(F):- formula(neg(F)).
formula(F1):- formula(or(F1,F2)).
formula(F2):- formula(or(F1,F2)).
formula(F1):- formula(until(Aut,Q,F1,F2)).
formula(F2):- formula(until(Aut,Q,F1,F2)).
formula(until(Aut,Q1,Alpha,Beta)):- formula(until(Aut,Q,Alpha,Beta)),
         trans(Aut,Q,Act,Q1).
formula(F1):- formula(and(F1,F2)).
formula(F2):- formula(and(F1,F2)).
formula(F1):- formula(impl(F1,F2)).
formula(F2):- formula(impl(F1,F2)).
formula(F):- formula(diamond(A,F)).
formula(F):- formula(ev(F)).
formula(F):- formula(box(A,F)).

% Encoding of Example 2

room(a).
room(b).

action(begin).
action(sense_mail(R)):- room(R).
action(deliver(R)):- room(R).
action(wait).

fluent(mail(R)):- room(R).

% action effects
```

```
holds(mail(R),SN):-
        room(R), occurs(sense_mail(R),S), SN=S+1,
        not -holds(mail(R),SN).
-holds(mail(R),SN):-
        room(R), occurs(deliver(R),S), SN=S+1.

% persistency

holds(F,SN):-
        holds(F,S),
        SN=S+1,
        not -holds(F,SN).
-holds(F,SN):-
        ~holds(F,S),
        SN=S+1,
        not holds(F,SN).

%preconditions

:- occurs(deliver(R),S), -holds(mail(R),S).
:- occurs(wait,S), holds(mail(R),S).

%initial state

holds(mail(R),0):- room(R), not -holds(mail(R),0).
-holds(mail(R),0):- room(R), not holds(mail(R),0).

% temporal constraints

formula(diamond(begin,true)).

:- not sat(diamond(begin,true),0).

formula(neg(ev(neg(box(begin,until(aut,q1,true,true)))))).

trans(aut,q1,sense_mail(a),q2).
trans(aut,q2,sense_mail(b),q3).
trans(aut,q3,deliver(a),q4).
trans(aut,q3,deliver(b),q4).
trans(aut,q3,wait,q4).
trans(aut,q4,begin,q5).
final(aut,q5).

:- not sat(neg(ev(neg(box(begin,until(aut,q1,true,true))))),0).
```

```
% counterexample (negated property)

formula(ev(neg(impl(mail(b),ev(neg(mail(b))))))).

:- not sat(ev(neg(impl(mail(b),ev(neg(mail(b)))))),0).
```

Appendix C

In this appendix we report tests of our approach for bounded model checking of DLTL formulas, in the line of the LTL BMC experiments in section 4 of (Heljanko and Niemelä 2003).

In particular, we consider the dining philosophers problems and the LTL formulas in section 4 of (Heljanko and Niemelä 2003); the relevant results are in Table 2 of that paper, columns $Int\ n$ and $Int\ s$, which provide, respectively, the smallest integer such that a counterexample of length n can be found using the interleaving semantics, and the time in seconds to find the counterexample. The interleaving semantics is the relevant one since in this paper we do not consider concurrent actions.

The general approach of the present paper can be directly mapped to the DLV-Complex extension of the DLV system, as shown in Appendix B. However, for a fairer comparison with the results in (Heljanko and Niemelä 2003), we have tested a representation of the dining philosophers problem, and of the LTL formulas to be verified, in the DLV system rather than in its DLV-Complex extension. Apart from not using parametric fluents and actions, this means that, rather than using clauses (in section 7) such as

$sat(or(Alpha, Beta), S) : -sat(Alpha, S).$

$sat(or(Alpha, Beta), S) : -sat(Beta, S).$

we provide, given the formula to be verified, a unique name for the formula and all its subformulas; and if a formula named $gamma$ is the disjunction $alpha \vee beta$ of formulas named $alpha$ and $beta$, we provide the clauses:

$sat(gamma, S) : -sat(alpha, S).$

$sat(gamma, S) : -sat(beta, S).$

and similarly for other operators. Such clauses can be easily generated automatically from the formula to be verified.

Moreover, we have applied some minor variation of the general approach in section 7 of our paper, such as using DLV built-in predicates.

Table C 1 reports the results obtained on a Dell PowerEdge server with 2 Intel Xeon E5520 processors (2.26Ghz, 8M Cache) and 32 Gb of memory.

Column n is the same as the $Int\ n$ column in Table 2 of (Heljanko and Niemelä 2003), i.e., the smallest integer such that a counterexample of length n can be found. Column “boundsmodels” is the analogous of the $Int\ s$ column in their paper (except that we include the result for 12 philosophers); it provides the running times in seconds to find a counterexample, running on our machine the code from http://www.tcs.hut.fi/kepa/experiments/boundsmodels/. The last column provides the running times in seconds to find a counterexample running in DLV the programs enclosed. The scalability of the approaches for such problems is similar, and this provides some evidence that the approaches have similar practical relevance for problems that can be represented easily in both of them.

Problem	n	boundsmodels	TemporalASP-DLV
DP(6)	8	0.1	0.1
DP(8)	10	1.4	2.4
DP(10)	12	29.1	115.7
DP(12)	14	7837.1	13036.2

Table C 1. *Dining philosophers results*

References

HELJANKO, K. AND NIEMELÄ, I. 2003. Bounded LTL model checking with stable models. *Theory and Practice of Logic Programming 3,* 4-5, 519–550.