TLP **17** (2): 226–243, 2017. © Cambridge University Press 2016
doi:10.1017/S1471068416000570 First published online 11 November 2016

Technical Note
*Temporal logic programs with variables**

FELICIDAD AGUADO, PEDRO CABALAR, GILBERTO PÉREZ and CONCEPCIÓN VIDAL

Dept. of Computer Science, Universidade da Coruña (University of Corunna), A Coruña, Spain
(e-mails: aguado@udc.es, cabalar@udc.es, gperez@udc.es, eicovima@udc.es)

MARTÍN DIÉGUEZ

IRIT - Université Paul Sabatier, Toulouse, France
(e-mail: martin.dieguez@irit.fr)

submitted 17 January 2016; revised 20 September 2016; accepted 23 September 2016

Abstract

In this note, we consider the problem of introducing variables in temporal logic programs under the formalism of *Temporal Equilibrium Logic*, an extension of Answer Set Programming for dealing with linear-time modal operators. To this aim, we provide a definition of a first-order version of Temporal Equilibrium Logic that shares the syntax of first-order Linear-time Temporal Logic but has different semantics, selecting some Linear-time Temporal Logic models we call *temporal stable models*. Then, we consider a subclass of theories (called *splittable temporal logic programs*) that are close to usual logic programs but allowing a restricted use of temporal operators. In this setting, we provide a syntactic definition of *safe variables* that suffices to show the property of *domain independence* – that is, addition of arbitrary elements in the universe does not vary the set of temporal stable models. Finally, we present a method for computing the derivable facts by constructing a non-temporal logic program with variables that is fed to a standard Answer Set Programming grounder. The information provided by the grounder is then used to generate a subset of ground temporal rules which is equivalent to (and generally smaller than) the full program instantiation.

KEYWORDS: artificial intelligence, knowledge representation, temporal logic, grounding, logic programming, answer set programming

1 Introduction

Many application domains and example scenarios from Answer Set Programming (ASP) (Marek and Truszczyński 1999; Niemelä 1999) contain a dynamic component,

* This research was partially supported by Spanish MINECO project TIN2013-42149-P and Xunta de Galicia GPC 2013/070.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

frequently representing transition systems over discrete time. In an attempt to provide a full logical framework for temporal reasoning in ASP, Aguado *et al.* (2008) proposed a formalism called *Temporal Equilibrium Logic* (TEL), syntactically identical to propositional *Linear-time Temporal Logic* (LTL) (Pnueli 1977), but semantically relying on a temporal extension of *Equilibrium Logic* (Pearce 1996), the most general and best studied logical characterisation of stable models (Gelfond and Lifschitz 1988). In Aguado *et al.* (2011), a reduction of (propositional) TEL into regular LTL was presented, but applicable to a subclass of temporal theories called *splittable Temporal Logic (STL) Programs*. This syntactic fragment deals with temporal rules in which, informally speaking, "past does not depend on the future", a restriction general enough to cover most (if not all) existing examples of ASP temporal scenarios. The reduction was implemented in a tool, `STeLP`[1] (Cabalar and Diéguez 2011), that computes the temporal stable models of a given program, that are shown as a Büchi automaton.

Although the approach in Aguado *et al.* (2011) was exclusively propositional, the input language of `STeLP` was extended with variables. As in non-temporal ASP, these were just understood as a shortcut for all their possible ground instances. This initial approach was not fully satisfactory for several reasons. First, it forced that any variable instance was not only *safe* (that is, occurring in the positive body of the rule) but also "typed" by a *static* predicate, i.e., a predicate whose extent does not vary along time. Second, this restriction implied sometimes the generation of irrelevant ground rules that increased the size of the resulting ground LTL theory whilst they could be easily detected and removed by a simple analysis of the temporal program. Last, but not least, the treatment of variables had no formal background and had not been proved to be sound with respect to the important property of *domain independence* (Bria *et al.* 2008) – essentially, a program is domain independent when its stable models do not vary under the arbitrary addition of new constants. Although the usual definition of safe variables guarantees domain independence, there was no formal proof for temporal logic programs under TEL.

In this note, we provide some results that allow an improved treatment of variables in temporal logic programs, using a first-order version of TEL as underlying logical framework. We relax the `STeLP` definition of safe variable by removing the need for static predicates so that, as in ASP, a variable in a rule is *safe* when it occurs in the positive body[2]. We prove that this simpler safety condition satisfies domain independence. Finally, we describe a method for grounding temporal logic programs under this new safety condition that still allows calling a standard ASP grounder as a backend, but using a positive normal logic program that is generated by a given transformation on the original temporal logic program.

[1] `http://kr.irlab.org/stelp_online`

[2] This definition of safety, initially introduced in `DLV` (Leone et al. 2006) has been adopted in the standard ASP-Core-2 (Calimeri *et al.* 2015) and also followed by `Gringo` (Gebser et al. 2011).

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

```
static city/1, car/1, road/2.

o at(X,A)  ::- driveto(X,A), car(X), city(A).                    % (1)

driveto(X,B) v no_driveto(X,B) ::- at(X,A), car(X), road(A,B).   % (2)

o at(X,A)    ::- at(X,A), not o no_at(X,A), car(X), city(A).     % (3)
  no_at(X,A) ::- at(X,B), A!=B, car(X), city(A), city(B).       % (4)

::- at(X,A), at(X,B), A!=B, car(X), city(A), city(B).            % (5)
```

Fig. 1. A simple car driving scenario.

2 A motivating example

Example 1

Suppose we have a set of cars placed at different cities and, at each transition, we can drive a car from one city to another in a single step, provided that there is a road connecting them.

Figure 1 contains a possible representation of this scenario in the language of STeLP. Operator "o" stands for "next" whereas "::-" corresponds to the standard ASP conditional ":-", but holding at all time points. Rule (1) is the effect axiom for driving car X to city A. The disjunctive rule (2) is used to generate possible occurrences of actions in a non-deterministic way. Rules (3) and (4) represent the inertia of fluent at(X,A). Finally, rule (5) forbids that a car is at two different cities simultaneously.

As we can see in the first line, predicates city/1, car/1 and road/2 are declared to be static. The scenario would be completed with rules for static predicates. These rules constitute what we call the *static program* and can only refer to static predicates without temporal operators. An example of a static program for this scenario could be

```
road(A,B) :- road(B,A).  % roads are bidirectional
city(A) :- road(A,B).
car(1). car(2).
road(lisbon,madrid). road(madrid,paris).
road(boston,ny). road(ny,nj).
```

Additionally, our temporal program would contain rules describing the initial state like, for instance, the pair of facts:

```
at(1,madrid). at(2,ny).
```

Note that all variables in a rule are always in some atom for a static predicate in the positive body. The current grounding process performed by STeLP just consists in feeding the static program to an ASP grounder (DLV or gringo) and, once it provides an extension for all the static predicates, each temporal rule is instantiated for each possible substitution of variables according to static predicates. In our

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

```
static city/1, car/1, road/2.

o at(X,A)  ::- driveto(X,A).

driveto(X,B) v no_driveto(X,B) ::- at(X,A), road(A,B).

o at(X,A)    ::- at(X,A), not o no_at(X,A).
  no_at(X,A) ::- at(X,B), A!=B, city(A).

::- at(X,A), at(X,B), A!=B.
```

Fig. 2. A possible variation of the cars scenario.

running example, for instance, the grounder provides a unique model[3] for the static program containing the facts:

```
car(1), car(2), city(lisbon), city(madrid), city(paris),
city(boston), city(ny), city(nj), road(lisbon,madrid),
road(madrid,lisbon), road(madrid,paris), road(paris,madrid),
road(boston,ny), road(ny,boston), road(ny,nj), road(nj,ny)
```

With these data, rule `(1)` generates 12 ground instances, since we have two possible cars for `X` and six possible cities for `A`. Similarly, rule `(4)` would generate 60 instances as there are 30 pairs `A,B` of different cities and two cars for `X`. Many of these ground rules, however, are irrelevant. Take, for instance,

```
o at(1,ny) ::- driveto(1,ny).
no_at(1,paris) ::- at(1,ny).
```

corresponding to possible instantiations of `(1)` and `(4)`, respectively. In both cases, the body refers to a situation where car 1 is located or will drive to New York, whilst we can observe that it was initially at Madrid and that the European roadmap is disconnected from the American one. Of course, one could additionally encode a static reachability predicate to force that rule instances refer to reachable cities for a given car, but this would not be too transparent or elaboration tolerant.

On the other hand, if we forget, for a moment, the temporal operators and we consider the definition of safe variables used in ASP, one may also wonder whether it is possible to simply require that each variable occurs in the positive body of rules, without needing to refer to static predicates mandatorily. Figure 2 contains a possible variation of the same scenario allowing this possibility. Our goal is allowing this new, more flexible definition of safe variables and exploiting, if possible, the information in the temporal program to reduce the set of generated ground rules.

[3] If the static program yields several stable models, each one generates a different ground theory whose temporal stable models are computed independently.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

3 Temporal quantified equilibrium logic

Syntactically, we consider function-free first-order languages $\mathscr{L} = \langle C, P\rangle$ built over a set of *constant* symbols, C, and a set of *predicate* symbols, P. Using $\mathscr{L}$, connectors and variables, an $\mathscr{L} = \langle C, P\rangle$-*formula* F is defined following the grammar:

$$F ::= p \mid \bot \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid F_1 \rightarrow F_2 \mid$$
$$\bigcirc F \mid \ \Box F \mid \Diamond F \mid \forall x F(x) \mid \exists x F(x)$$

where $p \in P$ is an atom, x is a variable and $\bigcirc$, $\Box$ and $\Diamond$, respectively stand for "next", "always" and "eventually". A *theory* is a finite set of formulas. We use the following derived operators:

$$\neg F \stackrel{\text{def}}{=} F \rightarrow \bot$$
$$\top \stackrel{\text{def}}{=} \neg\bot$$
$$F \leftrightarrow G \stackrel{\text{def}}{=} (F \rightarrow G) \wedge (G \rightarrow F)$$

for any formulas F, G. An *atom* is any $p(t_1, \ldots, t_n)$, where $p \in P$ is a predicate with n-arity and each t_i is a term (a constant or a variable) in its turn. We say that a term or a formula is *ground* if it does not contain variables. An $\mathscr{L}$-*sentence* or closed-formula is a formula without free-variables.

The application of i consecutive $\bigcirc$'s is denoted as follows: $\bigcirc^i\varphi \stackrel{\text{def}}{=} \bigcirc(\bigcirc^{i-1}\varphi)$ for $i > 0$ and $\bigcirc^0\varphi \stackrel{\text{def}}{=} \varphi$. A *temporal fact* is a construction of the form $\bigcirc^i A$, where A is an atom.

Let D be a non-empty set (the *domain* or *universe*). By $At(D, P)$, we denote the set of ground atomic sentences of the language $\langle D, P\rangle$. We will also define an interpretation σ of constants in C (and domain elements in D) as a mapping

$$\sigma : C \cup D \rightarrow D$$

such that $\sigma(d) = d$ for all $d \in D$.

A first-order LTL-interpretation is a structure $\langle (D, \sigma), \mathbf{T}\rangle$, where D and σ are as above and $\mathbf{T}$ is an infinite sequence of sets, $\mathbf{T} = \{T_i\}_{i\geqslant 0}$ with $T_i \subseteq At(D, P)$. Intuitively, T_i contains those ground atoms that are true at situation i. Given two LTL-interpretations $\mathbf{H}$ and $\mathbf{T}$ we say that $\mathbf{H}$ is *smaller than* $\mathbf{T}$, written $\mathbf{H} \leqslant \mathbf{T}$, when $H_i \subseteq T_i$ for all $i \geqslant 0$. As usual, $\mathbf{H} < \mathbf{T}$ stands for $\mathbf{H} \leqslant \mathbf{T}$ and $\mathbf{H} \neq \mathbf{T}$. We define the ground temporal facts associated to $\mathbf{T}$ as follows: $Facts(\mathbf{T}) \stackrel{\text{def}}{=} \{\bigcirc^i p \mid p \in T_i\}$. It is easy to see that $\mathbf{H} \leqslant \mathbf{T}$ iff $Facts(\mathbf{H}) \subseteq Facts(\mathbf{T})$.

Given $\mathbf{T}$ as above, we denote by $\mathbf{T}|_C$ the sequence of sets $\{T_i|_C\}_{i\geqslant 0}$, where each $T_i|_C = T_i \cap At(\sigma(C), P)$, i.e., those atoms from T_i that contain terms exclusively formed with universe elements that are images of syntactic constants in C.

Definition 1
A *temporal-here-and-there* $\mathscr{L}$-structure with static domains, or a **TQHT**-*structure*, is a tuple $\mathscr{M} = \langle (D, \sigma), \mathbf{H}, \mathbf{T}\rangle$, where $\langle (D, \sigma), \mathbf{H}\rangle$ and $\langle (D, \sigma), \mathbf{T}\rangle$ are two LTL-interpretations satisfying $\mathbf{H} \leqslant \mathbf{T}$.

A **TQHT**-structure of the form $\mathscr{M} = \langle (D, \sigma), \mathbf{T}, \mathbf{T}\rangle$ is said to be *total*. If $\mathscr{M} = \langle (D, \sigma), \mathbf{H}, \mathbf{T}\rangle$ is a **TQHT**-structure and k any positive integer, we denote by $(\mathscr{M}, k) =$

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

$\langle (D,\sigma),(\mathbf{H},k),(\mathbf{T},k)\rangle$ the temporal-here-and-there $\mathscr{L}$-structure with $(\mathbf{H},k)=\{H_i\}_{i\geqslant k}$ and $(\mathbf{T},k)=\{T_i\}_{i\geqslant k}$. The satisfaction relation for $\mathscr{M}=\langle (D,\sigma),\mathbf{H},\mathbf{T}\rangle$ is defined recursively forcing us to consider formulas from $\langle C\cup D,P\rangle$. Formally, if φ is an $\mathscr{L}$-sentence for the atoms in $At(C\cup D,P)$, then

- if $\varphi=p(t_1,\ldots,t_n)\in At(C\cup D,P)$, then

$$\mathscr{M}\models p(t_1,\ldots,t_n) \quad \text{iff} \quad p(\sigma(t_1),\ldots,\sigma(t_n))\in H_0.$$
$$\mathscr{M}\models t=s \quad \text{iff} \quad \sigma(t)=\sigma(s)$$

- $\mathscr{M}\not\models\bot$
- $\mathscr{M}\models\varphi\wedge\psi$ iff $\mathscr{M}\models\varphi$ and $\mathscr{M}\models\psi$.
- $\mathscr{M}\models\varphi\vee\psi$ iff $\mathscr{M}\models\varphi$ or $\mathscr{M}\models\psi$.
- $\mathscr{M}\models\varphi\rightarrow\psi$ iff $\langle (D,\sigma),w,\mathbf{T}\rangle\not\models\varphi$ or $\langle (D,\sigma),w,\mathbf{T}\rangle\models\psi$ for all $w\in\{\mathbf{H},\mathbf{T}\}$
- $\mathscr{M}\models\bigcirc\varphi$ if $(\mathscr{M},1)\models\varphi$.
- $\mathscr{M}\models\Box\varphi$ if $\forall j\geqslant 0$, $(\mathscr{M},j)\models\varphi$
- $\mathscr{M}\models\Diamond\varphi$ if $\exists j\geqslant 0$, $(\mathscr{M},j)\models\varphi$
- $\langle (D,\sigma),\mathbf{H},\mathbf{T}\rangle\models\forall x\varphi(x)$ iff $\langle (D,\sigma),w,\mathbf{T}\rangle\models\varphi(d)$ for all $d\in D$ and for all $w\in\{\mathbf{H},\mathbf{T}\}$.
- $\mathscr{M}\models\exists x\varphi(x)$ iff $\mathscr{M}\models\varphi(d)$ for some $d\in D$.

The resulting logic is called *quantified temporal here-and-there logic with static domains*, and denoted by **SQTHT** or simply by **QTHT**. It is not difficult to see that, if we restrict to total **TQHT**-structures, $\langle (D,\sigma),\mathbf{T},\mathbf{T}\rangle\models\varphi$ iff $\langle (D,\sigma),\mathbf{T},\mathbf{T}\rangle\models\varphi$ in first-order LTL. Furthermore, the following property can be easily checked by structural induction.

Proposition 1
For any formula φ, if $\langle (D,\sigma),\mathbf{H},\mathbf{T}\rangle\models\varphi$, then

$$\langle (D,\sigma),\mathbf{T},\mathbf{T}\rangle\models\varphi$$

A *theory* Γ is a set of $\mathscr{L}$-sentences. An interpretation $\mathscr{M}$ is a model of a theory Γ, written $\mathscr{M}\models\Gamma$, if it satisfies all the sentences in Γ.

Definition 2 (*Temporal equilibrium model*)
A *temporal equilibrium* model of a theory Γ is a total model $\mathscr{M}=\langle (D,\sigma),\mathbf{T},\mathbf{T}\rangle$ of Γ such that there is no $\mathbf{H}<\mathbf{T}$ satisfying $\langle (D,\sigma),\mathbf{H},\mathbf{T}\rangle\models\Gamma$.

If $\mathscr{M}=\langle (D,\sigma),\mathbf{T},\mathbf{T}\rangle$ is a temporal equilibrium model of a theory Γ, we say that the first-order LTL interpretation $\langle (D,\sigma),\mathbf{T}\rangle$ is a *temporal stable model* of Γ. We write $TSM(\Gamma)$ to denote the set of temporal stable models of Γ. The set of *credulous consequences* of a theory Γ, written $CredFacts(\Gamma)$ contains all the temporal facts that occur at some temporal stable model of Γ, that is

$$CredFacts(\Gamma)\stackrel{\text{def}}{=}\bigcup_{\langle (D,\sigma),\mathbf{T}\rangle\in TSM(\Pi)} Facts(\mathbf{T})$$

A property of TEL directly inherited from Equilibrium Logic (see Proposition 5 in Pearce (2006)) is the following:

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Proposition 2 (*Cumulativity for negated formulas*)
Let Γ be some theory and let $\neg\varphi$ be some formula such that $\mathscr{M} \models \neg\varphi$ for all temporal equilibrium models of Γ. Then, the theories Γ and $\Gamma \cup \{\neg\varphi\}$ have the same set of temporal equilibrium models.

In this work, we will further restrict the study to a syntactic subset called *splittable* temporal formulas (STF) which will be of one of the following types:

$$B \wedge N \rightarrow H \tag{3.1}$$

$$B \wedge \bigcirc B' \wedge N \wedge \bigcirc N' \rightarrow \bigcirc H' \tag{3.2}$$

$$\Box(B \wedge \bigcirc B' \wedge N \wedge \bigcirc N' \rightarrow \bigcirc H') \tag{3.3}$$

where B and B' are conjunctions of atomic formulas, N and N' are conjunctions of $\neg p$, being p an atomic formula and H and H' are disjunctions of atomic formulas.

Definition 3
An *STL-program* is a finite set of sentences like

$$\varphi = \forall x_1 \forall x_2 \ldots \forall x_n \psi$$

where ψ is an STF with $x_1, x_2, \ldots, x_n$ free variables.

We will also accept in an STL-program an implication of the form $\Box(B \wedge N \rightarrow H)$ (that is, containing $\Box$ but not any $\bigcirc$) understood as an abbreviation of the pair of STL-formulas:

$$B \wedge N \rightarrow H$$
$$\Box(\bigcirc B \wedge \bigcirc N \rightarrow \bigcirc H)$$

Example 2
The following theory Π_2 is an STL-program:

$$\neg p \rightarrow q \tag{3.4}$$

$$q \wedge \neg \bigcirc r \rightarrow \bigcirc p \tag{3.5}$$

$$\Box(q \wedge \neg \bigcirc p \rightarrow \bigcirc q) \tag{3.6}$$

$$\Box(r \wedge \neg \bigcirc p \rightarrow \bigcirc r \vee \bigcirc q) \tag{3.7}$$

For an example including variables, the encoding of Example 1 in Figure 2 is also an STL-program Π_1 whose logical representation corresponds to

$$\Box(\ Driveto(x,a) \rightarrow \bigcirc At(x,a)\ ) \tag{3.8}$$

$$\Box(\ At(x,a) \wedge Road(a,b) \rightarrow Driveto(x,b) \vee NoDriveto(x,b)\ ) \tag{3.9}$$

$$\Box(\ At(x,a) \wedge \neg \bigcirc NoAt(x,a) \rightarrow \bigcirc At(x,a)\ ) \tag{3.10}$$

$$\Box(\ At(x,b) \wedge City(a) \wedge a \neq b \rightarrow NoAt(x,a)\ ) \tag{3.11}$$

$$\Box(\ At(x,a) \wedge At(x,b) \wedge a \neq b \rightarrow \bot\ ) \tag{3.12}$$

Remember that all rule variables are implicitly universally quantified. For simplicity, we assume that inequality is a predefined predicate.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

An STL-program is said to be *positive* if for all rules (3.1)–(3.3), N and N' are empty (an empty conjunction is equivalent to $\top$). An STL-program is said to be *normal* if it contains no disjunctions, i.e., for all rules (3.1)–(3.3), H and H' are atoms.

Given a propositional combination φ of temporal facts with $\wedge, \vee, \bot, \rightarrow$, we denote φ^i as the formula resulting from replacing each temporal fact A in φ by $\bigcirc^i A$. For a formula $r = \Box\varphi$ like (3.3), we denote by r^i the corresponding φ^i. For instance, $(3.6)^i = (\bigcirc^i q \wedge \neg \bigcirc^{i+1} p \rightarrow \bigcirc^{i+1} q)$. As $\bigcirc$ behaves as a linear operator in THT, in fact $F^i \leftrightarrow \bigcirc^i F$ is a THT tautology.

Definition 4 (*expanded program*)
Given an STL-program Π for signature At we define its *expanded program* Π^∞ as the infinitary logic program containing all rules of the form (3.1), (3.2) in Π plus a rule r^i per each rule r of the form (3.3) in Π and each integer value $i \geqslant 0$.

The program Π_2^∞ consists of (3.4), (3.5) plus the infinite set of rules:

$$\bigcirc^i q \wedge \neg \bigcirc^{i+1} p \rightarrow \bigcirc^{i+1} q$$
$$\bigcirc^i r \wedge \neg \bigcirc^{i+1} p \rightarrow \bigcirc^{i+1} r \vee \bigcirc^{i+1} q$$

for $i \geqslant 0$. We can interpret the expanded program as an infinite, non-temporal program where the signature is the infinite set of atoms $\{\bigcirc^i p \mid p \in At,\ i \geqslant 0\}$.

Theorem 1 (*Theorem 1 in Aguado et al. (2011)*)
$\langle \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of Π iff $\{\bigcirc^i p \mid p \in T_i, i \geqslant 0\}$ is a stable model of Π^∞ under the (infinite) signature $\{\bigcirc^i p \mid p \in At\}$.

Proposition 3
Any normal positive STL-program Π has a unique temporal stable model $\langle (D, \sigma), \mathbf{T} \rangle$ which coincides with its $\leqslant$-least LTL-model. We denote $LM(\Pi) = Facts(\mathbf{T})$.

4 Safe variables and domain independence

In this section, we consider a definition of safe variables for temporal programs that removes the reference to static predicates.

Definition 5
An STF φ of type (3.1), (3.2) or (3.3) is said to be *safe* if, for any variable x occurring in φ, there exists an atomic formula p in B or B' such that x occurs in p. A formula $\forall x_1 \forall x_2 \ldots \forall x_n \psi$ is safe if the STF ψ is safe.

For instance, rules (3.8)–(3.12) are safe. A simple example of an unsafe rule is the STF:

$$\top \rightarrow p(x) \tag{4.1}$$

where x does not occur in the positive body. Although an unsafe rule does not always lead to lack of domain independence (see examples in Cabalar *et al.* (2009)), it is frequently the case. We prove next that domain independence is, in fact, guaranteed for safe STL-programs.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Theorem 2
If φ is a safe sentence and $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle$ is a temporal equilibrium model of φ, then $\mathbf{T}|_C = \mathbf{T}$ and $T_i \subseteq At(\sigma(C), P)$ for any $i \geqslant 0$.

Let (D,σ) be a domain and $D' \subseteq D$ a finite subset; the grounding over D' of a sentence φ, denoted by $\mathrm{Gr}_{D'}(\varphi)$, is defined recursively

$$\begin{aligned}
\mathrm{Gr}_{D'}(p) &\stackrel{\text{def}}{=} p, \text{ where } p \text{ denotes any atomic formula}\\
\mathrm{Gr}_{D'}(\varphi_1 \odot \varphi_2) &\stackrel{\text{def}}{=} \mathrm{Gr}_{D'}(\varphi_1) \odot \mathrm{Gr}_{D'}(\varphi_2),\\
&\quad \text{with } \odot \text{ any binary operator in } \{\wedge, \vee, \rightarrow\}\\
\mathrm{Gr}_{D'}(\forall x \varphi(x)) &\stackrel{\text{def}}{=} \bigwedge_{d \in D'} \mathrm{Gr}_{D'}\varphi(d)\\
\mathrm{Gr}_{D'}(\exists x \varphi(x)) &\stackrel{\text{def}}{=} \bigvee_{d \in D'} \mathrm{Gr}_{D'}\varphi(d)\\
\mathrm{Gr}_{D'}(\bigcirc \varphi) &\stackrel{\text{def}}{=} \bigcirc \mathrm{Gr}_{D'}(\varphi)\\
\mathrm{Gr}_{D'}(\Box \varphi) &\stackrel{\text{def}}{=} \Box \mathrm{Gr}_{D'}(\varphi)\\
\mathrm{Gr}_{D'}(\Diamond \varphi) &\stackrel{\text{def}}{=} \Diamond \mathrm{Gr}_{D'}(\varphi)
\end{aligned}$$

Theorem 3 (*Domain independence*)
Let φ be safe splittable temporal sentence. Suppose we expand the language $\mathscr{L}$ by considering a set of constants $C' \supseteq C$. A total **QTHT**-model $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle$ is a temporal equilibrium model of $\mathrm{Gr}_{C'}(\varphi)$ if and only if it is a temporal equilibrium model of $\mathrm{Gr}_C(\varphi)$.

5 Derivable ground facts

In this section, we present a technique for grounding safe temporal programs based on the construction of a positive normal ASP program with variables. The method is based on the idea of *derivable* ground temporal facts for an STL-program Π. This set, call it Δ, will be an upper estimation of the credulous consequences of the program, that is, $\mathit{CredFacts}(\Pi) \subseteq \Delta$. Of course, the ideal situation would be that $\Delta = \mathit{CredFacts}(\Pi)$, but the set $\mathit{CredFacts}(\Pi)$ requires the temporal stable models of Π and these (apart from being infinite sequences) will not be available at grounding time. In the worst case, we could choose Δ to contain the whole set of possible temporal facts, but this would not provide relevant information to improve grounding. So, we will try to obtain some superset of $\mathit{CredFacts}(\Pi)$ as small as possible, or if preferred, to obtain the largest set of *non-derivable* facts we can find. Note that a non-derivable fact $\bigcirc^i p \notin \Delta$ satisfies that $\bigcirc^i p \notin \mathit{CredFacts}(\Pi)$ and so, by Proposition 2, $\Pi \cup \{\neg \bigcirc^i p\}$ is equivalent to Π, that is, both theories have the same set of temporal equilibrium models. This information can be used to simplify the ground program either by removing rules or literals.

We begin defining several transformations on STL-programs. For any temporal rule r, we define $r^{\wedge}$ as the set of rules:

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

- If r has the form (3.1), then $r^{\wedge} \stackrel{\text{def}}{=} \{B \rightarrow p \mid \text{atom } p \text{ occurs in } H\}$.
- If r has the form (3.2), then $r^{\wedge} \stackrel{\text{def}}{=} \{B \wedge \bigcirc B' \rightarrow \bigcirc p \mid \text{atom } p \text{ occurs in } H'\}$.
- If r has the form (3.3), then $r^{\wedge} \stackrel{\text{def}}{=} \{\Box(B \wedge \bigcirc B' \rightarrow \bigcirc p) \mid \text{atom } p \text{ occurs in } H'\}$.

In other words, $r^{\wedge}$ will imply *all* the original disjuncts in the disjunctive head of r. It is interesting to note that for any rule r with an empty head ($\bot$), this definition implies $r^{\wedge} = \emptyset$. Program $\Pi^{\wedge}$ is defined as the union of $r^{\wedge}$ for all rules $r \in \Pi$. As an example, $\Pi_2^{\wedge}$ consists of the rules:

$$\begin{array}{lll} \top \rightarrow q & \Box(q \rightarrow \bigcirc q) & \Box(r \rightarrow \bigcirc r) \\ q \rightarrow \bigcirc p & & \Box(r \rightarrow \bigcirc q) \end{array}$$

whereas $\Pi_1^{\wedge}$ would be the program:

$$\Box(\ Driveto(x,a) \rightarrow \bigcirc At(x,a)\ ) \tag{5.1}$$

$$\Box(\ At(x,a) \wedge Road(a,b) \rightarrow Driveto(x,b)\ ) \tag{5.2}$$

$$\Box(\ At(x,a) \wedge Road(a,b) \rightarrow NoDriveto(x,b)\ ) \tag{5.3}$$

$$\Box(\ At(x,a) \rightarrow \bigcirc At(x,a)\ ) \tag{5.4}$$

$$\Box(\ At(x,b) \wedge City(a) \wedge a \neq b \rightarrow NoAt(x,a)\ ) \tag{5.5}$$

Notice that, by definition, $\Pi^{\wedge}$ is always a positive normal STL-program and, by Proposition 3, it has a unique temporal stable model, $LM(\Pi^{\wedge})$.

Proposition 4
For any STL-program Π, $CredFacts(\Pi) \subseteq LM(\Pi^{\wedge})$.

Unfortunately, using $\Delta = LM(\Pi^{\wedge})$ as set of derivable facts is unfeasible, since it contains infinite temporal facts corresponding to an "infinite run" of the transition system described by $\Pi^{\wedge}$. Instead, we will adopt a compromise solution taking a superset of $LM(\Pi^{\wedge})$ extracted from a new theory, Γ_{Π}. This theory will collapse all the temporal facts from situation 2 on, so that all the states T_i for $i \geqslant 2$ will be repeated[4]. We define Γ_{Π} as the result of replacing each rule $\Box(B \wedge \bigcirc B' \rightarrow \bigcirc p)$ in $\Pi^{\wedge}$ by the formulas:

$$B \wedge \bigcirc B' \rightarrow \bigcirc p \tag{5.6}$$

$$\bigcirc B \wedge \bigcirc^2 B' \rightarrow \bigcirc^2 p \tag{5.7}$$

$$\bigcirc^2 B \wedge \bigcirc^2 B' \rightarrow \bigcirc^2 p \tag{5.8}$$

and adding the axiom schema:

$$\bigcirc^2 \Box(p \leftrightarrow \bigcirc p) \tag{5.9}$$

for any ground atom $p \in At(D,P)$ in the signature of Π. As we can see, (5.6) and (5.7) are the first two instances of the original rule $\Box(B \wedge \bigcirc B' \rightarrow \bigcirc p)$ corresponding to situations $i = 0$ and $i = 1$. Formula (5.8), however, differs from the instance we

[4] Note that rules of the form (3.1) and (3.2) are not in the scope of $\Box$ and so may provide an irregular behaviour for atoms at situations 0 and 1. In a theory only consisting of rules like (3.3), we could collapse all situations from $i = 0$ on since they would follow a regular pattern.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

would get for $i = 2$ since, rather than having $\bigcirc^3 B'$ and $\bigcirc^3 p$, we use $\bigcirc^2 B'$ and $\bigcirc^2 p$, respectively. This can be done because axiom (5.9) is asserting that from situation 2 on all the states are repeated.

In the cars example, for instance, (5.1) from $\Pi_1^{\wedge}$ would yield the three rules:

$$Driveto(x,a) \rightarrow \bigcirc At(x,a)$$
$$\bigcirc Driveto(x,a) \rightarrow \bigcirc^2 At(x,a)$$
$$\bigcirc^2 Driveto(x,a) \rightarrow \bigcirc^2 At(x,a)$$

It is not difficult to see that axiom (5.9) implies that checking that some $\mathscr{M}$ is a temporal equilibrium model of Γ_Π is equivalent to checking that $\{\bigcirc^i p \mid p \in T_i, i = 0,1,2\}$ is a stable model of $\Gamma_\Pi \setminus \{(5.9)\}$ and fixing $T_i = T_2$ for $i \geqslant 3$. This allows us to exclusively focus on the predicate extents in T_0, T_1 and T_2, so we can see the $\Box$-free program $\Gamma_\Pi \setminus \{(5.9)\}$ as a positive normal ASP (i.e., non-temporal) program for the propositional signature $\{p, \bigcirc p, \bigcirc^2 p \mid p \in At(D,P)\}$ that can be directly fed to an ASP grounder, after some simple renaming conventions.

Theorem 4
Γ_Π has a least LTL-model, $LM(\Gamma_\Pi)$ which is a superset of $LM(\Pi^{\wedge})$.

In other words, $CredFacts(\Pi) \subseteq LM(\Pi^{\wedge}) \subseteq LM(\Gamma_\Pi) = \Delta$, i.e., we can use $LM(\Gamma_\Pi)$ as set of derivable facts and simplify the ground program accordingly. To this aim, a slight adaptation is further required. Each rule in Π like (3.3) has the form $\Box\alpha$ and any predicate p in α is *implicitly* affected (Theorem 1) by the extension of $\bigcirc^2 p$ in $LM(\Gamma_\Pi)$. In order to properly ground the extensions for $p, \bigcirc p$ and $\bigcirc^2 p$, we replace each $\Box\alpha$ by the equivalent conjunction of the three rules α, $\bigcirc\alpha$, and $\Box \bigcirc^2 \alpha$. For instance, (3.9) would be replaced by

$$At(x,a) \wedge Road(a,b) \rightarrow Driveto(x,b) \vee NoDriveto(x,b) \tag{5.10}$$
$$\bigcirc At(x,a) \wedge \bigcirc Road(a,b) \rightarrow \bigcirc Driveto(x,b) \vee \bigcirc NoDriveto(x,b) \tag{5.11}$$
$$\Box(\ \bigcirc^2 At(x,a) \wedge \bigcirc^2 Road(a,b) \rightarrow \bigcirc^2 Driveto(x,b) \vee \bigcirc^2 NoDriveto(x,b)\ ) \tag{5.12}$$

and then check the possible extents for the positive bodies we get from the set of derivable facts $\Delta = LM(\Gamma_\Pi)$. For example, for the last rule, we can make substitutions for x, a and b using the extents of $\bigcirc^2 At(x,a)$ and $\bigcirc^2 Road(a,b)$ we have in Δ. However, this still means making a join operation for both predicates. We can also use the ASP grounder for that purpose by just adding a rule that has as body, the positive body of the original temporal rule r, and as head, a new auxiliary predicate $Subst_r(x,a,b)$ referring to all variables in the rule. In the example, for rule (5.12), we would include in our ASP program:

$$\bigcirc^2 At(x,a) \wedge \bigcirc^2 Road(a,b) \rightarrow Subst_{(5.12)}(x,a,b)$$

In this way, each tuple of $Subst_r(x_1,\ldots,x_n)$ directly points out the variable substitution to be performed on the temporal rule.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

For instance, in the small instance case described of our example (two cars and six cities), we reduce the number of generated ground rules in the scope of "□" from 160 using the previous STeLP grounding method to 62. The reader may easily imagine that the higher degree of cities interconnection, the smaller obtained reduction of rule instances. Although an exhaustive experimentation is still ongoing work, a reduction of this kind is very promising. In our initial experiments, the grounding performed on Γ_Π (whose generation is polynomial) does not constitute a significant time increase, whereas the computation of temporal stable models is drastically improved by the reduction of ground rules[5].

6 Conclusions

We have improved the grounding method for temporal logic programs with variables in different ways. First, we provided a safety condition that directly corresponds to extrapolating the usual concept of safe variable in ASP. In this way, any variable occurring in a rule is considered to be safe if it also occurs in the positive body of the rule, regardless the possible scope of temporal operators and removing the previous dependence on the use of static predicates.

We have proved that this safety condition suffices to guarantee the property of *domain independence* by which computing the (temporal) stable models is insensitive to the possible addition of new arbitrary constants to the universe.

We have also designed a method for grounding the temporal logic program that consists in constructing a non-temporal normal positive program with variables that is fed to an ASP solver to directly obtain the set of variable substitutions to be performed for each rule. The proposed method allows reducing in many cases the number of ground temporal rules generated as a result.

The current note contains formal results, providing the correctness (with respect to domain independence) of the safety condition and the method for grounding safe programs. Regarding implementation, a stand-alone prototype for proving examples like the one in the paper has been constructed, showing promising results. The immediate next step is incorporating the new grounding method inside STeLP and analysing its performance on benchmark scenarios.

References

Aguado, F., Cabalar, P., Pérez, G. and Vidal, C. 2008. Strongly equivalent temporal logic programs. In *JELIA'08*, S. Hölldobler, C. Lutz and H. Wansing, Eds. Lecture Notes in Artificial Intelligence, vol. 5293. Springer, 8–20.

Aguado, F., Cabalar, P., Pérez, G. and Vidal, C. 2011. Loop formulas for splitable temporal logic programs. In *LPNMR'11*, J. P. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer, 80–92.

[5] The complexity of deciding whether a temporal stable model exists is EXPSPACE-complete in the general case (Bozzelli and Pearce 2015).

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Bozzelli, L. and Pearce, D. 2015. On the complexity of temporal equilibrium logic. In *Proc. of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science, (LICS'15)*. IEEE Computer Society, 645–656.

Bria, A., Faber, W. and Leone, N. 2008. Normal form nested programs. In *Proc. of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08)*, S. H. *et al*, Eds. Lecture Notes in Artificial Intelligence. Springer, 76–88.

Cabalar, P. and Diéguez, M. 2011. STELP - a tool for temporal answer set programming. In *LPNMR'11*, J. Delgrande and W. Faber, Eds. Lecture Notes in Artificial Intelligence, vol. 6645. Springer, 370–375.

Cabalar, P., Pearce, D. and Valverde, A. 2009. A revised concept of safety for general answer set programs. In *Proc. of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Lecture Notes in Computer Science, vol. 5753. Springer, 58–70.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F. and Schaub, T. 2015. ASP-core-2 input language format. Accessed 12 October 2016. URL: `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03c.pdf`.

Gebser, M., Kaminski, R., König, A. and Schaub, T. 2011. Advances in gringo series 3. In *Proc. of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, J. P. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer, 345–351.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming: Proc. of the 7th International Conference and Symposium*, R. A. Kowalski and K. A. Bowen, Eds., vol. 2. MIT Press, Cambridge, MA, 1070–1080.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2006. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7*, 499–562.

Marek, V. and Truszczyński, M. 1999. *Stable Models and an Alternative Logic Programming Paradigm*. Springer-Verlag, 169–181.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25*, 241–273.

Pearce, D. 1996. A new logical characterisation of stable models and answer sets. In *Proc. of the 2nd International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP'96)*, J. Dix, L. M. Pereira and T. C. Przymusinski, Eds. Lecture Notes in Artificial Intelligence, vol. 1216. Springer-Verlag, 1997, 57–60.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47* (1–2), 3–41.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 46–57.

van Emden, M. H. and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM 23*, 733–742.

Appendix. Proofs

Proof of Proposition 3

Consider the expanded program Π^{∞}. This is an infinitary positive (non-temporal) logic program. Using the well-known results by van Emden and Kowalski (1976), we know it has a least Herbrand model $LM(\Pi^{\infty})$ call it I, which may contain an infinite set of atoms in the signature of ground temporal facts $\{\bigcirc^i p \mid p \in At, i \geqslant 0\}$,

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

where At is the original signature of Π. Furthermore, as Π^∞ is positive, its unique stable model is precisely I. Given any set of ground temporal facts I, we can establish a one-to-one correspondence to an LTL-interpretation $\mathbf{I}$ with $Facts(\mathbf{I}) = I$. By Theorem 1, J is a stable model of Π^∞ iff $\mathbf{J}$, with $Facts(\mathbf{J}) = J$ is a temporal stable model of Π. Finally, as I is the unique stable model of Π^∞, we get that $\mathbf{I}$ is the only temporal stable model of Π.

A variable assignment μ in (D, σ) is a mapping from the set of variables to D. If $\varphi \in \mathscr{L}$ has free-variables, φ^μ is the closed formula obtained by replacing every free variable x by $\mu(x)$.

Lemma 1

Let φ be an STF and μ a variable assignment in (D, σ). If φ is safe, then if follows that:

$$\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle \models \varphi^\mu \text{ implies } \langle (D,\sigma), \mathbf{T}|_C, \mathbf{T}\rangle \models \varphi^\mu.$$

Proof

First of all, take $\varphi = B \wedge N \rightarrow H$ of type (3.1) and suppose that $\langle \mathbf{T}, \mathbf{T}\rangle \models \varphi^\mu$ but $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models \varphi^\mu$. This means that $\langle \mathbf{T}|_C, \mathbf{T}\rangle \models B^\mu \wedge N^\mu$ and $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models H^\mu$. Since $\langle \mathbf{T}, \mathbf{T}\rangle \models H^\mu$, there exists an atomic formula q in H such that $\langle \mathbf{T}, \mathbf{T}\rangle \models q^\mu$ but $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models q^\mu$. So we have a variable x in q with $\mu(x) \notin \sigma(C)$. As φ is safe, we know that x occurs in an atomic formula p in B. Then, $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models p^\mu$ and $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models B^\mu$ which yields a contradiction.

If φ is of type (3.2), we use a similar argument.

Finally, take $\varphi = \Box(B \wedge \bigcirc B' \wedge N \wedge \bigcirc N' \rightarrow \bigcirc H') = \Box\psi$ of type (3.3) and suppose that $\langle \mathbf{T}, \mathbf{T}\rangle \models \varphi^\mu$ but $\langle \mathbf{T}|_C, \mathbf{T}\rangle \not\models \varphi^\mu$. There exists $i \geqslant 0$ such that $\langle T_i, T_i\rangle \models \psi^\mu$ and $\langle T_i \cap \sigma(C), T_i\rangle \not\models \psi^\mu$. We then have that $\langle T_i \cap \sigma(C), T_i\rangle \models B^\mu \wedge (\bigcirc B')^\mu \wedge N^\mu \wedge (\bigcirc N')^\mu$ and $\langle T_i \cap \sigma(C), T_i\rangle \not\models (\bigcirc H')^\mu$. Using that φ is safe and the same argument as above, we find an atomic formula p in B or B' such that $\langle T_i \cap \sigma(C), T_i\rangle \not\models p^\mu$ which implies $\langle T_i \cap \sigma(C), T_i\rangle \not\models B^\mu \wedge (\bigcirc B')^\mu$ and leads to contradiction. The other implication follows directly from Proposition 1. □

Proposition 5

For any safe sentence $\varphi = \forall x_1 \forall x_2 \ldots \forall x_n \psi$

$$\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle \models \varphi \text{ iff } \langle (D,\sigma), \mathbf{T}|_C, \mathbf{T}\rangle \models \varphi.$$

Proof

Proceed by induction over the length of the prefix. If $n = 0$, we can take any μ assignment of variables and apply Lemma 1 on $\varphi = \varphi^\mu$. So take $\varphi = \forall x_1 \ldots \forall x_n \psi$ of length n and suppose that the result is true for any universal safe sentence whose prefix has length at most $n-1$. If $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle \models \varphi$, put $\varphi = \forall x_1 \alpha(x_1)$ with $\alpha(x_1) = \forall x_2 \ldots \forall x_n \psi$. For any $d \in D$, we know that $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle \models \alpha(d)$ and we have to show that $\langle (D,\sigma), \mathbf{T}|_C, \mathbf{T}\rangle \models \alpha(d)$. The induction hypothesis and the fact that $\alpha(d)$ is a safe sentence whose prefix has length smaller or equal than $n-1$ finishes the proof. □

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Proof of Theorem 2
If φ is a safe sentence and $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle$ is a temporal equilibrium model of φ, we have that $\langle (D,\sigma), \mathbf{T}|_C, \mathbf{T}\rangle \models \varphi$ by Proposition 5. The definition of temporal equilibrium model implies that $\mathbf{T}|_C = \mathbf{T}$ and $T_i \subseteq At(\sigma(C), P)$ for any $i \geqslant 0$.

Lemma 2
Let $\varphi(x)$ be a safe STF of type (3.1), (3.2) or (3.3) and take $\langle (D,\sigma), \mathbf{H}, \mathbf{T}\rangle$ be such that $\mathbf{T} = \mathbf{T}|_C$. Then, for any $d \in D \setminus \sigma(C)$, we have

$$\langle (D,\sigma), \mathbf{H}, \mathbf{T}\rangle \models \varphi(d).$$

Proof
First of all, suppose that $\varphi(x)$ is of type (3.1):

$$B \wedge N \rightarrow H$$

and take $d \in D \setminus \sigma(C)$ and $w \in \{\mathbf{H}, \mathbf{T}\}$ such that $\langle (D,\sigma), w, \mathbf{T}\rangle \not\models \varphi(d)$. This implies that $\langle (D,\sigma), w, \mathbf{T}\rangle \models B(d) \wedge N(d)$ but $\langle (D,\sigma), w, \mathbf{T}\rangle \not\models H(d)$. $\varphi(x)$ is safe so there must be an atom p in B such that x has an occurrence in p. Since $T_0 \subseteq \mathrm{At}(\sigma(C), P)$, it is clear that $\langle (D,\sigma), w, \mathbf{T}\rangle \not\models p(d)$, so $\langle (D,\sigma), w, \mathbf{T}\rangle \not\models B(d)$ which yields a contradiction.

The proof for the case of $\varphi(x)$ being of type (2) and (3) is similar. □

Lemma 3
Let $\varphi(x) = \forall x_1 \forall x_2 \ldots \forall x_n \psi$ with ψ a STF and such that $\varphi(x)$ has no other free variables than x. Let $\mathscr{M} = \langle (D,\sigma), \mathbf{H}, \mathbf{T}\rangle$ be such that $\mathbf{T} = \mathbf{T}|_C$. Then, if $\forall x \varphi(x)$ is safe, we have that

$$\mathscr{M} \models \forall x \varphi(x) \text{ iff } \mathscr{M} \models \textstyle\bigwedge_{c \in C} \varphi(c).$$

Proof
From left to right, just note that if $\mathscr{M} \models \forall x \varphi(x)$ but $\mathscr{M} \not\models \varphi(c)$, for some $c \in C$, we would have that $\mathscr{M} \not\models \varphi(\sigma(c))$ which would yield a contradiction.

For right to left, we can proceed by induction in n. If $n = 0$, then $\varphi(x)$ is in the case of the previous lemma for any $d \in D \setminus \sigma(C)$, so $\mathscr{M} \models \forall x \varphi(x)$ whenever $\mathscr{M} \models \bigwedge_{c \in C} \varphi(c)$. Now, suppose the result is true for any prenex formula with length up to $n-1$ and take $\varphi(x) = \forall x_1 \forall x_2 \ldots \forall x_n \psi(x, x_1, \ldots, x_n)$ such that $\mathscr{M} \models \bigwedge_{c \in C} \varphi(c)$. It only rests to show that $\mathscr{M} \models \varphi(d)$ for any $d \in D \setminus \sigma(C)$. Notice that $\varphi(d) = \forall x_1 \alpha(x_1)$ with $\alpha(x_1) = \forall x_2 \ldots \forall x_n \psi(d, x_1, x_2, \ldots, x_n)$. Since we can apply the induction hypothesis on $\alpha(x_1)$, it will be sufficient to prove that

$$\mathscr{M} \models \textstyle\bigwedge_{c \in C} \alpha(c).$$

Now, fix any $c \in C$ and take into account that

$$\mathscr{M} \models \varphi(c') = \forall x_1 \forall x_2 \ldots \forall x_n \psi(c', x_1, x_2 \ldots, x_n)$$

for all $c' \in C$, so we can replace x_1 by any constant in C, including c, and so

$$\mathscr{M} \models \forall x_2 \ldots \forall x_n \psi(c', c, x_2, \ldots, x_n), \text{ for any } c' \in C$$

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Observe that we can apply the induction hypothesis on $\beta(z)$, where

$$\beta(z) = \forall x_2 \ldots \forall x_n \psi(z, c, x_2, \ldots, x_n)$$

and then $\mathscr{M} \models \forall z \beta(z)$. In particular, $\mathscr{M} \models \beta(d)$ which completes the proof since $\beta(d) = \alpha(c)$. □

The following proposition can be easily proved

Proposition 6
Given any $D \neq \emptyset$: $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \varphi$ iff $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \mathrm{Gr}_D(\varphi)$.

Theorem 5
If $\varphi = \forall x_1 \forall x_2 \ldots \forall x_n \psi$ is a safe splittable temporal sentence and $\mathscr{M} = \langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle$ such that $\mathbf{T} = \mathbf{T}|_C$, then

$$M \models \varphi \text{ iff } \mathscr{M} \models \mathrm{Gr}_C(\varphi).$$

Proof
From left to right, suppose that $\mathscr{M} \models \varphi$. By Proposition 6, we know that $\mathscr{M} \models \mathrm{Gr}_D(\varphi)$. The result follows since $\sigma(C) \subseteq D$ and $\mathrm{Gr}_C(\varphi) = \mathrm{Gr}_{\sigma(C)}(\varphi)$.

Now, from the right to left direction, take $\varphi = \forall x_1 \forall x_2 \ldots \forall x_n \psi$ a safe splittable temporal sentence and suppose that $\mathscr{M} \models \mathrm{Gr}_C(\varphi)$. Again, we can proceed by induction in n. If $n = 0$, then φ is quantifier free so $\mathrm{Gr}_C(\varphi) = \varphi$. Suppose the result is true for any safe splittable sentence with length up to $n-1$ and put $\varphi = \forall x_1 \alpha(x_1)$ with $\alpha(x_1) = \forall x_2 \ldots \forall x_n \psi(x_1, x_2, \ldots, x_n)$. Notice that $\alpha(x_1)$ is a safe formula that has no more free variables than x_1, so, if we apply Lemma 3, it will be sufficient to show that $\mathscr{M} \models \bigwedge_{c \in C} \alpha(c)$. Since we are supposing that

$$\mathscr{M} \models \mathrm{Gr}_C(\varphi) = \bigwedge_{c \in C} \mathrm{Gr}_C(\alpha(c)),$$

and we can apply the induction hypothesis on any $\alpha(c)$ with $c \in C$, it follows that $\mathscr{M} \models \bigwedge_{c \in C} \alpha(c)$ and this completes the proof. □

Theorem 6
If φ is a safe splittable temporal sentence, then $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of φ iff $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of $\mathrm{Gr}_C(\varphi)$.

Proof
Suppose that $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of φ and $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \mathrm{Gr}_C(\varphi)$. Since φ is safe, we know by Theorem 2 that $\mathbf{T} = \mathbf{T}|_C$ so, applying Theorem 5, it follows that $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \varphi$ and $\mathbf{H} = \mathbf{T}$. This shows that $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is also a temporal equilibrium model of $\mathrm{Gr}_C(\varphi)$, The other implication follows directly from the fact that $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \varphi$ implies $\langle (D, \sigma), \mathbf{H}, \mathbf{T} \rangle \models \mathrm{Gr}_C(\varphi)$. □

Proof of Theorem 3
Let us show that the following assertions are equivalent:

(1) $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of $\mathrm{Gr}_C(\varphi)$;
(2) $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of φ;
(3) $\langle (D, \sigma), \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of $\mathrm{Gr}_{C'}(\varphi)$.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

Taking into account the previous theorem, we only have to prove the equivalence of 2 and 3. Suppose that $\langle (D,\sigma), \mathbf{T}, \mathbf{T}\rangle$ is a temporal equilibrium model of φ and $\langle (D,\sigma), \mathbf{H}, \mathbf{T}\rangle \models \mathrm{Gr}_{C'}(\varphi)$. Because of Theorem 2, we have that $\mathbf{T} = \mathbf{T}|_C \subseteq \mathbf{T}|_{C'}$ and an obvious extension of Theorem 5 to C', implies that

$$\langle (D,\sigma), \mathbf{H}, \mathbf{T}\rangle \models \varphi$$

and so $\mathbf{H} = \mathbf{T}$. This shows that 2 implies 3. The other implication (3. $\Longrightarrow$ 2.) follows directly.

Lemma 4
If T is any equilibrium model of a (non-temporal) program Π with rules of type (3.1), then $T \subseteq J$, where J is any model of the normal positive program $\Pi^{\wedge}$.

Proof
We will prove that $\langle T \cap J, T\rangle \models \Pi$, and so, $T \cap J = T$ by the minimality of T.

Let $B \wedge N \rightarrow H$ of the form (3.1) be an arbitrary rule in Π. To prove $\langle T \cap J, T\rangle \models r$, we already know that $\langle T, T\rangle \models r$ and remain to prove that if $\langle T \cap J, T\rangle \models B \wedge N$, then $\langle T \cap J, T\rangle \models H$. So, suppose that $\langle T \cap J, T\rangle \models B \wedge N$. Then, $\langle T, T\rangle \models B \wedge N$ y $\langle J, J\rangle \models B$. Therefore, $\langle T, T\rangle \models H$ and there exists $p \in H$ such that $\langle T, T\rangle \models p$. Since rule $B \rightarrow p \in \Pi^{\wedge}$ and $\langle J, J\rangle \models B$, we get that $\langle J, J\rangle \models p$ and so $\langle T \cap J, T\rangle \models H$, as we wanted to prove. □

Given any rule like r like (3.2) of (3.3) and a set of atoms X, we define its *simplification* $simp(r, X)$ as

$$simp(r,X) \stackrel{\text{def}}{=} \begin{cases} \bigcirc B' \wedge \bigcirc N' \rightarrow \bigcirc H' & \text{if } B \subseteq X \text{ and } N \cap X = \emptyset \\ \top & \text{otherwise} \end{cases}$$

Definition 6 (*Slice program*)
Given some LTL interpretation $\mathbf{T}$, let us define now the sequence of programs:

$$slice(\Pi, \mathbf{T}, 0) \stackrel{\text{def}}{=} \Pi^0 = ini_0(\Pi)$$
$$slice(\Pi, \mathbf{T}, 1) \stackrel{\text{def}}{=} \{simp(r, T_0) \mid r \in ini_1(\Pi) \cup dyn(\Pi)\}$$
$$slice(\Pi, \mathbf{T}, i+1) \stackrel{\text{def}}{=} \{\bigcirc^i simp(r, T_i) \mid r \in dyn(\Pi)\} \qquad \text{for } i \geqslant 1$$

Theorem 7 (*Theorem 3 in (Aguado et al. 2011)*)
Let $\langle \mathbf{T}, \mathbf{T}\rangle$ be a model of a splittable TLP Π. $\langle \mathbf{T}, \mathbf{T}\rangle$ is a temporal equilibrium model of Π iff

(i) $\mathbf{T}^0 = T_0$ is a stable model of $slice(\Pi, \mathbf{T}, 0) = \Pi^0 = ini_0(\Pi)$ and
(ii) $(\mathbf{T}^1 \setminus At^0)$ is a stable model of $slice(\Pi, \mathbf{T}, 1)$ and
(iii) $(\mathbf{T}^i \setminus At^{i-1})$ is a stable model of $slice(\Pi, \mathbf{T}, i)$ for $i \geqslant 2$.

Proof of Proposition 4
Let $\langle \mathbf{T}, \mathbf{T}\rangle$ be any temporal equilibrium model of Π an denote by $\{L_i\}_{i \geqslant 0}$ the corresponding infinite sequence of ground atoms of $LM(\Pi^{\wedge})$. By Theorem 7, we know that, for all $i \geqslant 0$, T_i (resp. L_i) is a stable model of $slice(\Pi, \mathbf{T}, i)$

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press

(resp. of $slice(\Pi^{\wedge}, LM(\Pi^{\wedge}), i)$. Finally, we can apply Lemma 4 and the fact that $slice(\Pi, \mathbf{T}, 0)^{\wedge} = slice(\Pi^{\wedge}, LM(\Pi^{\wedge}), 0)$ and, for $i \geqslant 1$,

$$slice(\Pi, \mathbf{T}, i)^{\wedge} \subseteq slice(\Pi^{\wedge}, LM(\Pi^{\wedge}), i).$$

Proof of Theorem 4

Let $\langle \mathbf{T}, \mathbf{T} \rangle$ be the unique temporal equilibrium model of $\Pi^{\wedge}$ and let $\langle \mathbf{D}, \mathbf{D} \rangle$ denote the temporal interpretation defined by

- $D_i = T_i$ if $0 \leqslant i \leqslant 1$,
- D_2 is the stable model of the positive non-disjunctive program:

$$\{\bigcirc^2 B \wedge \bigcirc^2 B' \rightarrow \bigcirc^2 p \mid \Box(B \wedge \bigcirc B' \rightarrow \bigcirc p) \in dyn(\Pi^{\wedge})\} \cup slice(\Pi^{\wedge}, \mathscr{L}, 1),$$

- $D_i = D_2$ if $\geqslant 3$.

It is straightforward to check that $\langle \mathbf{D}, \mathbf{D} \rangle$ is a temporal equilibrium model of Γ_{Π}. Notice that $T_2 \subseteq D_2$. This follows from Lemma 4 and the facts that $\mathbf{T}^2 \setminus AT^1$ is the stable model of $slice(\Pi^{\wedge}, \mathscr{L}, 1)$ and D_2 is a model of this latter program.

The cases $i = 0, 1, 2$ follow from Proposition 4 in the fact that $T_2 \subseteq D_2$.

When $i \geqslant 3$, we shall prove that $\langle \mathbf{T}^i \setminus At^{i-1} \cap \mathbf{D}^i \setminus At^{i-1}, \mathbf{T}^i \setminus At^{i-1} \rangle$ is a model of $slice(\Pi^{\wedge}, \mathbf{T}, i)$ so by Theorem 7, $\mathbf{T}^i \setminus At^{i-1} \cap \mathbf{D}^i \setminus At^{i-1} = \mathbf{T}^i \setminus At^{i-1}$ and, consequently, $T_i \subseteq D_i$. So, take $\bigcirc^i B' \rightarrow \bigcirc^i H' \in slice(\Pi^{\wedge}, \mathbf{T}, i)$ and suppose that

$$\langle \mathbf{T}^i \setminus At^{i-1} \cap \mathbf{D}^i \setminus At^{i-1}, \mathbf{T}^i \setminus At^{i-1} \rangle \models \bigcirc^i B'$$

This fact implies that $\langle \mathbf{T}^i \setminus At^{i-1}, \mathbf{T}^i \setminus At^{i-1} \rangle \models \bigcirc^i B'$ and also that there exists a (positive normal) dynamic rule like (3.3) such that $B \subseteq T_{i-1} \subseteq D_{i-1}$. Since $\langle \mathbf{T}^i \setminus At^{i-1}, \mathbf{T}^i \setminus At^{i-1} \rangle$ is a model of $slice(\Pi^{\wedge}, \mathbf{T}, i)$, the only atom $p \in H'$ satisfies $\langle \mathbf{T}^i \setminus At^{i-1}, \mathbf{T}^i \setminus At^{i-1} \rangle \models \bigcirc^i p$. It only rests to show that $\langle \mathbf{D}^i, \mathbf{D}^i \rangle \models \bigcirc^i p$ or equivalently $\langle \mathbf{D}, \mathbf{D} \rangle \models \bigcirc^2 p$ (notice that $D_i = D_2$ if $i \geqslant 2$). Finally, we can use that the rule $\bigcirc^2 B \wedge \bigcirc^2 B' \rightarrow \bigcirc^2 p \in \Gamma_{\Pi}$ and also the fact that $\langle \mathbf{D}, \mathbf{D} \rangle \models \bigcirc^2 B \wedge \bigcirc^2 B'$ because $i \geqslant 3$ and $B' \subseteq D_i = D_2$ and $B \subseteq T_{\imath-1} \subseteq D_{i-1} = D_2$.

https://doi.org/10.1017/S1471068416000570 Published online by Cambridge University Press