Journal of Applied Non-Classical Logics
Vol. 00, No. 00, Month 201X, 1–24

RESEARCH ARTICLE

Temporal Equilibrium Logic: a survey

Felicidad Aguado, Pedro Cabalar, Martín Diéguez,
Gilberto Pérez and Concepción Vidal

Department of Computer Science, University of Corunna (Spain)
{aguado,cabalar,martin.dieguez,gperez,eicovima}@udc.es

(*Received 00 Month 201X; final version received 00 Month 201X*)

This paper contains a survey of the main definitions and results obtained up to date related to Temporal Equilibrium Logic, a non-monotonic hybrid approach that combines Equilibrium Logic (the best-known logical characterisation for the stable models semantics of logic programs) with Linear-Time Temporal Logic.

Keywords: Answer Set Programming, Linear Temporal Logic, Equilibrium Logic, Nonmonotonic Reasoning, Knowledge Representation.

1. Introduction

After three decades of research in Nonmonotonic Reasoning (NMR), the paradigm of *Answer Set Programming* (ASP) [33, 30, 3] has been consolidated as one of the most prominent and successful approaches for practical knowledge representation. Great part of this success is due to the impressive advances in implementation of efficient solvers and development of applications. In this aspect, ASP shows some similarities to SAT (propositional satisfiability), since it is also targeted for solving problems in the NP complexity class (at least when we restrict to non-disjunctive ASP) and there already exists a regular ASP solver competition [12]. However, unlike SAT, a crucial factor in ASP which is also responsible for its success is its versatility for knowledge representation. While most applications of SAT involve an external translation of the problem to solve into clauses in propositional logic, ASP provides a powerful language based on logic program rules with variables, allowing non-monotonicity through default negation and offering additional expressive features such as disjunction in the head, aggregates, constraints on numeric variables, functions, etc. Furthermore, all these syntactic enhancements rely on a simple semantics, *stable models* [21] (or *answer sets*), that has eventually become a kind of convergence point for different NMR formalisms and approaches (see the survey [27]).

Despite of all these features, there exists one important aspect in knowledge representation that has not been included among the usual ASP language extensions: *temporal reasoning*. Temporal scenarios are very frequent in ASP applications and benchmarks while many action languages have been defined as front-ends for ASP [22, 13, 20, 19].

This research was partially supported by Spanish MEC project TIN2009-14562-C05-04.

In fact, ASP offers essential features for a suitable formal representation of temporal scenarios. For instance, it provides a high degree of *elaboration tolerance*[1] [31] allowing a simple and natural solution to typical representational issues such as the *frame* problem and the *ramification* problem (see respectively [32] and [25]). Another interesting feature is that it allows a uniform treatment of different kinds of reasoning problems such as prediction, postdiction, planning, diagnosis or verification. However, since ASP is not a temporal formalism, it also involves some difficulties for dealing with temporal problems. For instance, as most ASP tools must deal with finite domains, this additionally requires fixing a finite plan length[2] with an obvious impossibility for solving problems such as proving the non-existence of a plan for a given planning scenario or verifying temporal properties of the transition system behaviour. In principle, one may think that this kind of problems dealing with unbound time are typically best suited for modal temporal logics, whose expressive power, computation methods (usually decidable) and associated complexity have been extensively well-studied. Unfortunately, as happens with SAT in the non-temporal case, temporal logics are not designed for knowledge representation. For instance, the best known temporal logics are monotonic, so that the frame and ramification problems constantly manifest in their applications, even for very simple scenarios.

From the previous discussion, it seems that a reasonable choice would be trying to get the advantages from both worlds: keeping all the nice knowledge representation features from ASP while taking benefit from the advances in the area of modal temporal logic. A crucial result that makes this combination possible is the logical characterisation of stable models in terms of David Pearce's *Equilibrium Logic* [34, 35]. While many of the definitions of stable models described in [27] are very interesting for their practical understanding, not all of them are so convenient when we are interested in a purely logical formalism. Equilibrium Logic has been proved to be a powerful tool for the theoretical analysis of ASP, motivating the study of strong equivalence[1] between logic programs [28], covering most syntactic extensions considered up to date, or being closely related to the conception of new definitions of stable models for arbitrary propositional [15] and first order theories [17]. Another important advantage is that its formal definition is extremely simple: it amounts to selecting a kind of minimal models among those obtained with the (monotonic) intermediate logic of *Here-and-There* (HT) [23].

Using Equilibrium Logic as a starting point, the definition of new extended or hybrid logics becomes quite straigthforward. In [11], Cabalar and Pérez-Vega proposed a formalism called *Temporal Equilibrium Logic* (TEL) which shares the syntax of propositional *Linear-time Temporal Logic* (LTL)[24, 36] but extends the semantics of Equilibrium Logic for temporal paths of infinite length. In this way, as happened in Equilibrium Logic, TEL models are just the result of a kind of minimisation among models of the monotonic logic of *Temporal Here-and-There* (THT), a combination of HT and LTL. Of course, for the choice of a temporal extension, many other combinations can be imagined. However, it makes sense to start with one of the simplest and best-known temporal logics which, on the other hand, suffices by far for the temporal reasoning required in most ASP benchmarks dealing with dynamic scenarios.

This paper contains a survey of the main definitions of TEL and the most relevant results obtained up to date, together with open topics for future work. The rest of the paper is organised as follows. In the next section we introduce the logic of THT

[1] In McCarthy's words: "A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances."

[2] In a similar technique to planning as propositional satisfiability [26].

[1] Two programs are strongly equivalent when they yield the same stable models even when they are included in a common larger program or context.

(Temporal Here-and-There) which constitutes the monotonic basis of TEL. This section also contains an alternative three-valued characterisation plus a pair of transformations to encode LTL in THT and vice versa. Section 3 contains the definition of temporal equilibrium models as a kind of minimisation among THT-models of a theory. Next, we discuss in Section 4 the property of strong equivalence and a possible method to check it. In Section 5 we present a normal form which is closer to non-temporal ASP logic programs. Section 6 presents two methods for computing temporal equilibrium models. The first one is more efficient, but accepts only a syntactic subset of the normal form described in the previous section. The second method is applicable to any arbitrary theory, but is less efficient in its current implementation. Finally, Section 7 concludes the paper.

2. Temporal Here-and-There

The logic of *Linear Temporal Here-and-There* (THT) is defined as follows. We start from a finite set of atoms At called the *propositional signature*. A (temporal) *formula* F is defined by the grammar:

$$F ::= \bot \mid p \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid F_1 \rightarrow F_2 \mid \bigcirc F_1 \mid F_1 \,\mathcal{U}\, F_2 \mid F_1 \,\mathcal{R}\, F_2$$

where F_1 and F_2 are temporal formulae in their turn and p is any atom. The unary temporal operator $\bigcirc$ is read as "next," and the binary temporal operators $\mathcal{U}$ and $\mathcal{R}$ are read as "until" and "release" respectively. A formula is said to be *non-modal* if it does not contain temporal operators. Negation is defined as $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$ whereas $\top \stackrel{\text{def}}{=} \neg\bot$. As usual, $\varphi \leftrightarrow \psi$ stands for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. Other usual temporal operators can be defined in terms of $\mathcal{U}$ and $\mathcal{R}$ as follows:

$$\Box\varphi \stackrel{\text{def}}{=} \bot \,\mathcal{R}\, \varphi \qquad \Diamond\varphi \stackrel{\text{def}}{=} \top \,\mathcal{U}\, \varphi$$

$\Box$ is read "forever" and $\Diamond$ stands for "eventually" or "at some future point." We define the following notation for a finite concatenation of $\bigcirc$'s

$$\bigcirc^0\varphi \stackrel{\text{def}}{=} \varphi$$
$$\bigcirc^i\varphi \stackrel{\text{def}}{=} \bigcirc(\bigcirc^{i-1}\varphi) \quad (\text{with } i \geq 1)$$

The semantics of the logic of THT is defined in terms of sequences of pairs of propositional interpretations. A (temporal) *interpretation* $\mathbf{M}$ is an infinite sequence of pairs $m_i = \langle H_i, T_i\rangle$ with $i = 0, 1, 2, \ldots$ where $H_i \subseteq T_i$ are sets of atoms standing for *here* and *there* respectively. For simplicity, given a temporal interpretation, we write $\mathbf{H}$ (resp. $\mathbf{T}$) to denote the sequence of pair components $H_0, H_1, \ldots$ (resp. $T_0, T_1, \ldots$). Using this notation, we will sometimes abbreviate the interpretation as $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$. An interpretation $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$ is said to be *total* when $\mathbf{H} = \mathbf{T}$.

Definition 1 (THT-Satisfaction)**.** *The satisfaction relation* $\models$ *is interpreted as follows on THT models (*$\mathbf{M}$ *is a THT model and* $k \in \mathbb{N}$*):*

(1) $\mathbf{M}, k \models p$ *iff* $p \in H_k$*, for any atom* $p \in At$*.*
(2) $\mathbf{M}, k \models \varphi \wedge \psi$ *iff* $\mathbf{M}, k \models \varphi$ *and* $\mathbf{M}, k \models \psi$*.*
(3) $\mathbf{M}, k \models \varphi \vee \psi$ *iff* $\mathbf{M}, k \models \varphi$ *or* $\mathbf{M}, k \models \psi$*.*
(4) $\mathbf{M}, k \models \varphi \rightarrow \psi$ *iff for all* $\mathbf{H}' \in \{\mathbf{H}, \mathbf{T}\}$*,* $\langle \mathbf{H}', \mathbf{T}\rangle, k \not\models \varphi$ *or* $\langle \mathbf{H}', \mathbf{T}\rangle, k \models \psi$*.*

(5) $\mathbf{M}, k \models \bigcirc\varphi$ *iff* $\mathbf{M}, k+1 \models \varphi$.
(6) $\mathbf{M}, k \models \varphi \,\mathcal{U}\, \psi$ *iff there is* $j \geq k$ *such that* $\mathbf{M}, j \models \psi$ *and for all* $j' \in [k, j-1]$, $\mathbf{M}, j' \models \varphi$.
(7) $\mathbf{M}, k \models \varphi \,\mathcal{R}\, \psi$ *iff for all* $j \geq k$ *such that* $\mathbf{M}, j \not\models \psi$, *there exists* $j' \in [k, j-1]$, $\mathbf{M}, j' \models \varphi$.
(8) never $\mathbf{M}, k \models \bot$. ⊠

A formula φ is THT-*valid* if $\mathbf{M}, 0 \models \varphi$ for any $\mathbf{M}$. An interpretation $\mathbf{M}$ is a THT-*model* of a theory Γ, written $\mathbf{M} \models \Gamma$, if $\mathbf{M}, 0 \models \varphi$, for all formula $\varphi \in \Gamma$. An axiomatisation for THT is still an open topic.

We assume that a finite sequence $\mathbf{M} = m_1, m_2, \ldots, m_n$ is an abbreviation of an infinite sequence where the remaining elements coincide with m_n, that is, that for $i > n$, $m_i = m_n$. The logic of THT is an orthogonal combination of the logic of HT and the (standard) linear temporal logic (LTL). When we disregard temporal operators, we obtain the logic of HT. On the other hand, if we restrict the semantics to total interpretations, $\langle \mathbf{T}, \mathbf{T} \rangle \models \varphi$ corresponds to satisfaction of formulas $\mathbf{T} \models \varphi$ in LTL.

2.1 *THT versus LTL*

As we can see in Definition 1, the main difference with respect to LTL is the interpretation of implication (item 4), that must be checked in both components, $\mathbf{H}$ and $\mathbf{T}$, of $\mathbf{M}$. In fact, as we said before, it is easy to see that when we take total models $\mathbf{M} = \langle \mathbf{T}, \mathbf{T} \rangle$, THT satisfaction $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$ collapses to standard LTL satisfaction $\mathbf{T}, k \models \varphi$ so that we will sometimes write the latter when convenient. For instance, item 4 in Definition 1 can be rewritten as:

4′. $\mathbf{M}, k \models \varphi \to \psi$ iff ($\mathbf{M}, k \models \varphi$ implies $\mathbf{M}, k \models \psi$) and $\mathbf{T}, k \models \varphi \to \psi$ (LTL satisfaction)

A result inherited from HT whose proof can be obtained by structural induction is the so-called *persistence* property.

Proposition 2 (Persistence; see [5])**.** *For any formula* φ, *any THT model* $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ *and any* $i \geq 0$, *if* $\mathbf{M}, i \models \varphi$, *then* $\mathbf{T}, i \models \varphi$. ⊠

A consequence of this proposition is that the interpretation of negation $\neg\varphi$ in $\langle \mathbf{H}, \mathbf{T} \rangle$ amounts to checking that φ does not hold in the total model $\mathbf{T}$. Formally:

Corollary 3. $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \neg\varphi$ *iff* $\mathbf{T}, i \not\models \varphi$ *in LTL.* ⊠

Obviously, any THT valid formula is also LTL valid, but not the other way around. For instance, the following are THT valid equivalences:

$$\neg(\varphi \wedge \psi) \leftrightarrow \neg\varphi \vee \neg\psi \tag{1}$$

$$\neg(\varphi \vee \psi) \leftrightarrow \neg\varphi \wedge \neg\psi \tag{2}$$

$$\bigcirc(\varphi \oplus \psi) \leftrightarrow \bigcirc\varphi \oplus \bigcirc\psi \tag{3}$$

$$\bigcirc \otimes \varphi \leftrightarrow \otimes \bigcirc \varphi \tag{4}$$

$$\varphi \,\mathcal{U}\, \psi \leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathcal{U}\, \psi)) \tag{5}$$

$$\varphi \,\mathcal{R}\, \psi \leftrightarrow \psi \wedge (\varphi \vee \bigcirc(\varphi \,\mathcal{R}\, \psi)) \tag{6}$$

for any binary connective $\oplus$ and any unary connective $\otimes$. Equivalences (1),(2) mean that De Morgan laws are valid whereas (3),(4) allow us to shift the '$\bigcirc$' operator to all the operands of any connective. Formulas (5) and (6) provide inductive definitions

for "until" and "release" respectively. The following result captures a general class of LTL-valid formulas that are also THT-valid.

Proposition 4 (from [5]). *Let φ and ψ be two formulas not containing implication*[1]*. Then $\varphi \leftrightarrow \psi$ is THT-valid iff it is LTL-valid.* ⊠

There are, however, LTL-valid formulas that are not THT-valid. As an example, the formula $\varphi \vee \neg\varphi$ (known as *excluded middle* axiom) is not THT valid. This feature is inherited from the intermediate/intuitionistic nature of HT. In fact, the addition of this axiom makes THT collapse to LTL, much in the same way as it makes intuitionistic logic collapse to classical propositional logic. The following proposition shows that if we add a copy of this axiom for any atom at any position of the models, we can force THT models of any formula to be total.

Proposition 5 (from [6]). *Given a temporal formula φ for a propositional signature At, for every THT model $\langle \mathbf{H}, \mathbf{T}\rangle$, the propositions below are equivalent:*

(I) $\langle \mathbf{H}, \mathbf{T}\rangle, 0 \models \varphi \wedge \bigwedge_{p\in At} \Box(p \vee \neg p)$,
(II) $\mathbf{T}, 0 \models \varphi$ *in LTL, and for $i \geq 0$ and $p \in At$, we have $p \in H_i$ iff $p \in T_i$.* ⊠

This gives us a direct way of encoding LTL in THT, since LTL models of φ coincide with its total THT models.

The translation from THT to LTL is not so straightforward. Informally speaking, it requires adding an auxiliary atom p' for each atom p in the signature, so that the former captures the truth at component $\mathbf{H}$ in a THT model $\langle \mathbf{H}, \mathbf{T}\rangle$ while the latter represents truth at $\mathbf{T}$. Formally, given a propositional signature At, let us denote $At^* = At \cup \{p' \mid p \in At\}$ which is going to be the new propositional signature in LTL. For any temporal formula φ we define its translation φ^* as follows:

(1) $\bot^* \stackrel{\text{def}}{=} \bot$
(2) $p^* \stackrel{\text{def}}{=} p'$ for any $p \in \Sigma$
(3) $(\bigcirc\varphi)^* \stackrel{\text{def}}{=} \bigcirc\varphi^*$
(4) $(\varphi \oplus \psi)^* \stackrel{\text{def}}{=} \varphi^* \oplus \psi^*$ for any binary operator $\oplus \in \{\wedge, \vee, \mathcal{U}, \mathcal{R}\}$
(5) $(\varphi \rightarrow \psi)^* \stackrel{\text{def}}{=} (\varphi \rightarrow \psi) \wedge (\varphi^* \rightarrow \psi^*)$

From the last point and the fact that $\neg\varphi = \varphi \rightarrow \bot$, it follows that $(\neg\varphi)^* = (\varphi \rightarrow \bot) \wedge (\varphi^* \rightarrow \bot) = \neg\varphi \wedge \neg\varphi^*$. Similarly, $(\varphi \leftrightarrow \psi)^* = (\varphi \leftrightarrow \psi) \wedge (\varphi^* \leftrightarrow \psi^*)$ and $(\Box\varphi)^* = \Box\varphi^*$, $(\Diamond\varphi)^* = \Diamond\varphi^*$.

We associate to any THT interpretation $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$ the LTL interpretation $\mathbf{M}^t = \mathbf{I}$ in LTL defined as the sequence of sets of atoms $\mathbf{I} = \{I_i\}_{i\in\mathbb{N}}$ where $I_i = \{p' \mid p \in H_i\} \cup T_i$. As a THT interpretation must satisfy $H_i \subseteq T_i$ by construction, we may have LTL interpretations that do not correspond to any THT one. In particular, for an arbitrary $\mathbf{I}$, we will only be able to form some $\mathbf{M}$ such that $\mathbf{M}^t = \mathbf{I}$ when the set of primed atoms at each I_i is a subset of the non-primed ones. In other words, only LTL interpretations $\mathbf{I}$ satisfying the axiom schema:

$$\Box(p' \rightarrow p) \tag{7}$$

for any atom $p \in At$ will have a corresponding THT interpretation M such that $\mathbf{I} = \mathbf{M}^t$.

Example 6. $\mathbf{M} = ((\emptyset, \{p, q\}), (\{p\}, \{p, q\}), (\{q\}, \{q\}))$ *is a THT model of the theory $\{\Box(\neg p \rightarrow q) \wedge \Diamond q\}$. In the same way, the corresponding LTL interpretation $\mathbf{M}^t$ =*

[1] Remember that negation is a form of implication.

$(\{p,q\},\{p',p,q\},\{q',q\})$ is an LTL model of

$$\begin{aligned}(\Box(\neg p \to q) \wedge \Diamond q)^* &\leftrightarrow \Box(\neg p \to q)^* \wedge (\Diamond q)^* \\ &\leftrightarrow \Box((\neg p \to q) \wedge ((\neg p)^* \to q')) \wedge \Diamond q' \\ &\leftrightarrow \Box((\neg p \to q) \wedge ((\neg p \wedge \neg p') \to q')) \wedge \Diamond q'.\end{aligned}$$

Theorem 7 (from [2]). *Let $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$ be any THT interpretation and φ any formula. For any $i \geq 0$, it holds that*

(a) *$\langle \mathbf{H}, \mathbf{T}\rangle, i \models \varphi$ if and only if $\mathbf{M}^t, i \models \varphi^*$ in LTL; and*
(b) *$\langle \mathbf{T}, \mathbf{T}\rangle, i \models \varphi$ if and only if $\mathbf{M}^t, i \models \varphi$ in LTL.* ⊠

Corollary 8 (from [6]). *Let φ' be the formula $\varphi^* \wedge \bigwedge_{p \in At} \Box(p' \to p)$. Then the set of LTL models for the formula φ' corresponds to the set of THT models for the temporal formula φ.* ⊠

2.2 *A Three-valued characterisation of THT*

The intermediate logic of HT can be seen as a three-valued logic (in fact, it corresponds to Gödel's logic G_3). The intuition behind this correspondence is that, given an HT interpretation $\langle H, T\rangle$ with $H \subseteq T$ sets of atoms, we can have three possible situations for any atom p: $p \in H$ (the atom is true), $p \notin T$ (the atom is false) or $p \in T \setminus H$ (the atom is undefined). We can extend this same idea to the case of THT defining an alternative three-valued characterisation of this logic by seeing each $m_i = \langle H_i, T_i\rangle$ as a three-valued mapping $m_i : At \to \{0,1,2\}$. Thus, for any atom p, $m_i(p) = 0$ when $p \notin T_i$ (the atom is false), $m_i(p) = 2$ when $p \in H_i$ (the atom is true), and $m_i(p) = 1$ when $p \in T_i \setminus H_i$ (the atom is undefined). We can then define a valuation for any formula φ at time point i, written[1] $\mathbf{M}(i, \varphi)$, by similarly considering which formulas are satisfied by $\langle \mathbf{H}, \mathbf{T}\rangle$ (which will be assigned 2), not satisfied by $\langle \mathbf{T}, \mathbf{T}\rangle$ (which will be assigned 0) or none of the two (which will take value 1). From the definitions in the previous section, we can easily derive the following conditions:

(1) $\mathbf{M}(i,p) \stackrel{\text{def}}{=} m_i(p)$

(2) $\mathbf{M}(i, \varphi \wedge \psi) \stackrel{\text{def}}{=} min(\mathbf{M}(i,\varphi), \mathbf{M}(i,\psi)); \quad \mathbf{M}(i, \varphi \vee \psi) \stackrel{\text{def}}{=} max(\mathbf{M}(i,\varphi), \mathbf{M}(i,\psi))$

(3) $\mathbf{M}(i, \varphi \to \psi) \stackrel{\text{def}}{=} \begin{cases} 2 & \text{if } \mathbf{M}(i,\varphi) \leq \mathbf{M}(i,\psi) \\ \mathbf{M}(i,\psi) & \text{otherwise} \end{cases}$

(4) $\mathbf{M}(i, \bigcirc\varphi) \stackrel{\text{def}}{=} \mathbf{M}(i+1, \varphi)$

(5) $\mathbf{M}(i, \varphi \,\mathcal{U}\, \psi) \stackrel{\text{def}}{=} \begin{cases} 2 \text{ if } \exists j \geq i : \mathbf{M}(j,\psi) = 2 \text{ and } \forall k, i \leq k < j \Rightarrow \mathbf{M}(k,\varphi) = 2 \\ 0 \text{ if } \forall j \geq i : \mathbf{M}(j,\psi) = 0 \text{ or } \exists k, i \leq k < j, \mathbf{M}(k,\varphi) = 0 \\ 1 \text{ otherwise} \end{cases}$

(6) $\mathbf{M}(i, \varphi \,\mathcal{R}\, \psi) \stackrel{\text{def}}{=} \begin{cases} 2 \text{ if } \forall j \geq i : \mathbf{M}(j,\psi) = 2 \text{ or } \exists k, i \leq k < j, \mathbf{M}(k,\varphi) = 2 \\ 0 \text{ if } \exists j \geq i : \mathbf{M}(j,\psi) = 0 \text{ and } \forall k, i \leq k < j \Rightarrow \mathbf{M}(k,\varphi) = 0 \\ 1 \text{ otherwise} \end{cases}$

From their definition, the interpretation of the temporal derived operators becomes $\mathbf{M}(i, \Box\varphi) = min\ \{\mathbf{M}(j,\varphi) \mid j \geq i\}$ and $\mathbf{M}(i, \Diamond\varphi) = max\ \{\mathbf{M}(j,\varphi) \mid j \geq 0\}$.

[1] We use the same name $\mathbf{M}$ for a temporal interpretation and for its induced three-valued valuation function – ambiguity is removed by the way in which it is applied (a structure or a function on indices and formulas).

Under this alternative three-valued definition, an interpretation $\mathbf{M}$ *satisfies* a formula φ when $\mathbf{M}(0,\varphi) = 2$. When $\mathbf{M} = \langle \mathbf{T}, \mathbf{T} \rangle$, its induced valuation will be just written as $\mathbf{T}(i,\varphi)$ and obviously becomes a two-valued function, that is $\mathbf{T}(i,\varphi) \in \{0,2\}$.

3. Temporal Equilibrium Models

We can now proceed to describe the model selection criterion that defines temporal equilibrium models. Given two interpretations $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ and $\mathbf{M'} = \langle \mathbf{H'}, \mathbf{T'} \rangle$ we say that $\mathbf{M'}$ is *lower or equal than* $\mathbf{M}$, written $\mathbf{M'} \leq \mathbf{M}$, when $\mathbf{T'} = \mathbf{T}$ and for all $i \geq 0$, $H'_i \subseteq H_i$. As usual, $\mathbf{M'} < \mathbf{M}$ stands for $\mathbf{M'} \leq \mathbf{M}$ and $\mathbf{M'} \neq \mathbf{M}$.

Definition 9 (Temporal Equilibrium Model)**.** *An interpretation* $\mathbf{M}$ *is a* temporal equilibrium model *of a theory* Γ *if* $\mathbf{M}$ *is a total model of* Γ *and there is no other* $\mathbf{M'} < \mathbf{M}$*, such that* $\mathbf{M'} \models \Gamma$*.* ⊠

Note that any temporal equilibrium model is total, that is, it has the form $\langle \mathbf{T}, \mathbf{T} \rangle$ and so can be actually seen as an LTL-interpretation of the form $\mathbf{T}$ that we will call *temporal stable model*.

Definition 10 (Temporal Stable Model)**.** *If* $\langle \mathbf{T}, \mathbf{T} \rangle$ *is a temporal equilibrium model of a theory* Γ *then* $\mathbf{T}$ *is called a* temporal stable model *of* Γ*.* ⊠

In this way, the fact that any temporal stable model of Γ is also an LTL-model becomes trivial.

Note that the consequence relation induced by temporal equilibrium models is non-monotonic. In fact, when we restrict the syntax to ASP programs and the semantics to HT interpretations of the form $\langle H_0, T_0 \rangle$ we talk about (non-temporal) equilibrium models, which coincide with stable models in their most general definition [15]. The result below establishes a more general relation to non-temporal equilibrium logic/ASP.

Proposition 11 (from [11])**.** *Let* Γ *be a combination of non-modal connectives* $\wedge, \vee, \neg, \rightarrow, \bot$*, with their usual grammar, with expressions like* $\bigcirc^i p$*, being* p *an atom, and let* n *be the maximum value for* i *in all* $\bigcirc^i p$ *occurring in* Γ*. Then* $\langle \mathbf{T}, \mathbf{T} \rangle$ *is a temporal equilibrium model of* Γ *iff (1)* $T_i = \emptyset$ *for all* $i > n$ *; and (2)* $\langle X, X \rangle$ *with* $X = \bigcup_{i=0}^{n} \{\bigcirc^i p \mid p \in T_i\}$ *is an equilibrium model of* Γ*, reading each '*$\bigcirc^i p$*' as a new atom in the signature.* ⊠

The TEL satisfiability problem consists in determining whether a temporal formula has a TEL model.

3.1 *Examples*

As a first example, consider the formula

$$\Box(\neg p \rightarrow \bigcirc p) \tag{8}$$

Its intuitive meaning corresponds to the logic program consisting of rules of the form: $p(s(X)) \leftarrow \mathit{not}\ p(X)$ where time has been reified as an extra parameter $X = 0, s(0), s(s(0)), \ldots$. Notice that the interpretation of $\neg$ is that of default negation *not* in logic programming. In this way, (8) is saying that, at any situation, if there is no evidence on p, then p will become true in the next state. In the initial state, we have no evidence on p, so this will imply $\bigcirc p$. To derive $\bigcirc \bigcirc p$ the only possibility would be the rule $\neg \bigcirc p \rightarrow \bigcirc \bigcirc p$, an instance of (8). As the body of this rule is false, $\bigcirc \bigcirc p$ becomes false by default, and so on. It is easy to see that the unique temporal stable

model of (8) is captured by the formula $\neg p \wedge \Box(\neg p \leftrightarrow \bigcirc p)$.

As a second example, take the formula $\Diamond p$. This formula informally corresponds to an infinite disjunction $p \vee \bigcirc p \vee \bigcirc \bigcirc p \vee \ldots$. Again, as happens in disjunctive logic programming, in TEL we have a truth minimality condition that will make true the formula with as little information as possible. As a result, it is easy to see that the temporal stable models of $\Diamond p$ are captured by the formula $\neg p \,\mathcal{U}\, (p \wedge \bigcirc\Box\neg p)$ whose models are those where p holds true at exactly one position.

It is worth noting that an LTL satisfiable formula may have no temporal stable model. As a simple example (well-known from non-temporal ASP) the logic program rule $\neg p \rightarrow p$, whose only (classical) model is $\{p\}$, has no stable models. If we assume that p cannot be derived, i.e. $\neg p$, then the rule contradicts the assumption. On the other hand, if we assume that p can be derived, then $\neg p$ becomes false and we are left with no rule that *justifies* a possible derivation for p.

When dealing with logic programs, it is well-known that non-existence of stable models is always due to a kind of cyclic dependence on default negation like this. In the temporal case, however, non-existence of temporal stable models may also be due to a lack of a finite justification for satisfying the criterion of minimal knowledge. As an example, consider the formula:

$$\Box(\neg\bigcirc p \rightarrow p) \wedge \Box(\bigcirc p \rightarrow p) \tag{9}$$

This formula has no temporal equilibrium models. To see why, note that (9) is LTL-equivalent (and THT-equivalent) to $\Box(\neg \bigcirc p \vee \bigcirc p \rightarrow p)$ that, in its turn, is LTL-equivalent to $\Box p$. Thus, the only LTL-model $\mathbf{T}$ of 9 has the form $T_i = \{p\}$ for any $i \geq 0$. However, it is easy to see that the interpretation $\langle \mathbf{H}, \mathbf{T}\rangle$ with $H_i = \emptyset$ for all $i \geq 0$ is also a THT model, whereas $\mathbf{H} < \mathbf{T}$. It is worth to note that (9) was extracted from a first-order counterpart, the pair of rules $\neg p(s(X)) \rightarrow p(X)$ and $p(s(X)) \rightarrow p(X)$, that were used in [14] to show that an acyclic[1] program without a well-founded dependence ordering relation may have no stable models. In this case, we accordingly get no temporal stable models.

Another example of TEL-unsatisfiable formula is $\Box\Diamond p$, typically used in LTL to assert that property p occurs infinitely often. This formula has no temporal stable models: all models must contain infinite occurrences of p and there is no way to establish a minimal $\mathbf{H}$ among them. Thus, formula $\Box\Diamond p$ is LTL satisfiable but it has no temporal stable model. This formula is normally used in LTL to specify a *liveness* property (p occurs infinitely often). When asserted in TEL, however, this yields a conflict with the minimality criterion: informally speaking, for an infinite set of p's along time, we can always take a smaller model by removing one p. The result would also be an infinite set of p's. So, there is no way to get a minimal model. This does not mean a serious limitation in expressiveness, since for practical problems, we would usually include this type of liveness property in a constraint, rather than asserting the formula. In this way, the constraint $\neg\Box\Diamond p \rightarrow \bot$ would be ruling out all models where p does not occur intinitely often. Note also that all LTL is embeddable both in THT (Proposition 5) and in fact in TEL too (see [6] for details).

By contrast, the next proposition states that for a large class of temporal formulas, LTL satisfiability is equivalent to THT satisfiability and TEL satisfiability.

Proposition 12 (from [6]). *Let φ be temporal formula built over the connectives $\vee$, $\wedge$, $\rightarrow$, $\bigcirc$ and $\mathcal{U}$ and such that $\rightarrow$ occurs only in subformulae of the form $p \rightarrow \bot$ with*

[1] A logic program is *acyclic* if its corresponding dependency graph contains no cycles. This graph has as vertices the set of atoms in the program and one edge (p, q) for each rule with p in the head and q occurring in the rule body.

$p \in At$. The propositions below are equivalent: (I) φ is LTL satisfiable; (II) φ is THT satisfiable; (III) φ has a temporal stable model, i.e. φ is TEL satisfiable.

Theorem 13 (Corollary 2 from [6]). *THT satisfiability problem is* PSPACE*-complete.*

4. Strong equivalence

By $Eq(V, \Gamma)$ we denote the set of temporal equilibrium models under signature V of a theory $\Gamma \subseteq \mathcal{L}_V$. Remember that the consequence relation induced by temporal equilibrium models is nonmonotonic. Thus, when dealing with equivalence of two theories, Γ_1, Γ_2, the mere coincidence of equilibrium models $Eq(V, \Gamma_1) = Eq(V, \Gamma_2)$ will not suffice for safely replacing one by each other, since they may behave in a different way in the presence of additional information. Two theories Γ_1, Γ_2 are said to be *strongly equivalent* when $Eq(V, \Gamma_1 \cup \Gamma) = Eq(V, \Gamma_2 \cup \Gamma)$ for any arbitrary theory Γ.

The following result shows that we can use translation $(\cdot)^*$ from THT to LTL to obtain a sufficient condition for TEL-strong equivalence of two temporal theories.

Theorem 14 (Main theorem from [2]). *Let Γ_1 and Γ_2 be a pair of temporal theories, and $\bigwedge \Gamma_1$ and $\bigwedge \Gamma_2$ the conjunctions of their respective sets of formulas. Then Γ_1 and Γ_2 are strongly equivalent with respect to temporal equilibrium models if the formula $\bigwedge_{p \in At} \Box(p' \rightarrow p) \rightarrow (\bigwedge \Gamma_1 \leftrightarrow \bigwedge \Gamma_2)^*$ is valid in LTL.*

We may also use this result to detect a redundant formula φ in some theory Γ. To this aim, we would have to show that Γ and $\Gamma' = \Gamma \setminus \{\varphi\}$ are strongly equivalent. From the theorem above, it follows that:

Corollary 15. *Let Γ be a temporal theory, $\bigwedge \Gamma$ the conjunction of its formulas and φ some arbitrary temporal formula. Then Γ and $\Gamma \cup \{\varphi\}$ are strongly equivalent if the formula $\bigwedge_{p \in At} \Box(p' \rightarrow p) \rightarrow (\bigwedge \Gamma \rightarrow \varphi)^*$ is valid in LTL.* ⊠

In [2], Theorem 14 was used to compare a pair of example theories for the same actions domain, checking whether their respective representations allowed the removal of inertia laws. This technique was automated into a prototype called `tht`[1].

5. Normal Form

Normal forms are usually interesting for building computation methods for a given logical formalism. For instance, in the case of Equilibrium Logic, it has been already proved [8] that any arbitrary propositional theory is strongly equivalent to a logic program (allowing disjunction and negation in the head). In this way, logic programs constitute a normal form for Equilibrium Logic. Similarly, in the case of (monotonic) LTL, an implicational clause-like normal form introduced in [18] was used for designing a temporal resolution method.

Following [5], in this section we show that TEL can be similarly reduced (under strong equivalence) to a normal form, called *temporal logic programs* (TLP), consisting of a set of implications (embraced by a necessity operator) quite close to logic program rules. The reduction into normal form starts from the structure-preserving polynomial transformation presented in [9] for the non-temporal case. This transformation has as a main feature the introduction of an auxiliary atom per each subformula in the original theory. We then combine this technique with the inductive definitions of temporal operators

[1] Available at `http://equilibriumlogic.irlab.org`

used for LTL in [18]. The obtained normal form considerably reduces the possible combinations of modal operators and, as we will see later, has become useful for a practical computation of TEL models.

5.1 *Temporal Logic Programs*

Next, we describe the normal form we are interested in. Given a signature At, we define a *temporal literal* as any expression in the set $\{p, \bigcirc p, \neg p, \neg \bigcirc p \mid p \in At\}$.

Definition 16 (Temporal rule)**.** *A* temporal rule *is either:*

(1) an initial rule *of the form*

$$B_1 \wedge \cdots \wedge B_n \rightarrow C_1 \vee \cdots \vee C_m \tag{10}$$

where all the B_i and C_j are temporal literals, $n \geq 0$ and $m \geq 0$.

(2) a dynamic rule *of the form $\Box r$, where r is an initial rule.*

(3) a fulfillment rule *like $\Box(\Box p \rightarrow q)$ or like $\Box(p \rightarrow \Diamond q)$ with p, q atoms.* ⊠

In the three cases, the antecedent and consequent of the (unique) implication receive the names of rule *body* and rule *head* respectively. In initial (resp. dynamic) rules, we may have an empty head $m = 0$ corresponding to $\bot$ – if so, we talk about an *initial* (resp. *dynamic*) *constraint*. A *temporal logic program*[1] (TLP for short) is a finite set of temporal rules. A TLP without temporal operators, that is, a set of initial rules without $\bigcirc$, is said to be an *ASP program*[2].

As an example of TLP take the program Π_1 consisting of:

$$\neg a \wedge \bigcirc b \rightarrow \bigcirc a \tag{11}$$

$$\Box(a \rightarrow b) \tag{12}$$

$$\Box(\neg b \rightarrow \bigcirc a) \tag{13}$$

where (11) is an initial rule and (12),(13) are dynamic rules.

Looking at the semantics of $\Box$ it seems clear that we can understand a dynamic rule $\Box r$ as an infinite sequence of expressions like $\bigcirc^i r$, one for each $i \geq 0$. Using (3),(4) we can shift $\bigcirc^i$ inside all connectives in r so that $\bigcirc^i r$ is equivalent to an initial rule resulting from prefixing any atom in r with $\bigcirc^i$. To put an example, if $r =$ (13) then $\bigcirc^2 r$ would correspond to $(\neg \bigcirc^2 b \rightarrow \bigcirc^3 a)$.

Definition 17 (i-expansion of a rule)**.** *Given $i \geq 0$, the* i-expansion *of a dynamic rule $\Box r$, written $(\Box r)^i$, is a set of rules defined as:*

$$(\Box r)^i \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{if } i = 0 \text{ and } r \text{ contains some } `\bigcirc' \\ \{\bigcirc^j r \mid 0 \leq j \leq i-1\} & \text{if } i > 0 \text{ and } r \text{ contains some } `\bigcirc' \\ \{\bigcirc^j r \mid 0 \leq j \leq i\} & \text{otherwise} \end{cases}$$

If r is an initial rule, its i-expansion is defined as:

$$r^i \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{if } i = 0 \text{ and } r \text{ contains some } `\bigcirc' \\ r & \text{otherwise} \end{cases}$$ ⊠

[1] In fact, as shown in [5], this normal form can be even more restrictive: initial rules can be replaced by atoms, and we can avoid the use of literals of the form $\neg \bigcirc p$.

[2] In ASP literature, this is called a a disjunctive program with negation in the head.

In this way, the superindex i refers to the longest sequence of $\bigcirc$'s used in the rule. For instance, $(13)^3$ would be:

$$\{\ \ (\neg b \to \bigcirc a),\ \ (\neg\bigcirc b \to \bigcirc^2 a),\ \ (\neg\bigcirc^2 b \to \bigcirc^3 a)\ \ \}$$

We extend this notation to programs, so that given a TLP Π its i-expansion Π^i results from replacing each initial or dynamic rule r in Π by r^i. An interesting observation is that we can understand each Π^i as a (non-temporal) ASP program for signature $At^i \stackrel{\text{def}}{=} \{\text{“}\bigcirc^j p\text{”} \mid p \in At, 0 \leq j \leq i\}$ where we understand each “$\bigcirc^j p$” as a different propositional atom. This same notation can be applied to interpretations. If $\mathbf{T}$ is an LTL interpretation (an infinite sequence of sets of atoms) for signature At its i-expansion would be the corresponding propositional interpretation for signature At^i defined as $\mathbf{T}^i \stackrel{\text{def}}{=} \{\bigcirc^j p \mid 0 \leq j \leq i, p \in T_j\}$ and if $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$ is a THT interpretation then its i-expansion is defined as the HT interpretation $\mathbf{M}^i \stackrel{\text{def}}{=} \langle \mathbf{H}^i, \mathbf{T}^i\rangle$. In all these cases, we also define the ω-*expansion* (or simply, *expansion*) as the infinite union of all i-expansions for all $i \geq 0$. Thus, for instance $(\Box r)^\omega \stackrel{\text{def}}{=} \bigcup_{i\geq 0}(\Box r)^i$ and similarly for Π^ω, At^ω, $\mathbf{T}^\omega$ and $\mathbf{M}^\omega$. It is interesting to note that, for any classical interpretation $\mathbf{T}'$ for signature At^ω, we can always build a corresponding LTL interpretation $\mathbf{T}$ in signature At such that $\mathbf{T}^\omega = \mathbf{T}'$. The following theorem establishes the correspondence between a temporal program and its expansion.

Theorem 18. *Let* Π *be a TLP without fulfillment rules. Then* $\langle \mathbf{T}, \mathbf{T}\rangle$ *is a temporal equilibrium model of* Π *under signature At iff* $\mathbf{T}^\omega$ *is a stable model of* Π^ω *under signature* At^ω. ⊠

The above theorem allows us reading a TLP with initial and dynamic rules as an ASP program with infinite “copies” of the same rule schemata. In many cases, this allows us to foresee the temporal equilibrium models of a TLP. For instance, if we look at our example TLP Π_1, it is easy to see that we should get $T_0 = \emptyset$ as the only rule affecting the situation $i = 0$ is $(12)^0 = (a \to b)$. For situation $i = 1$ we would have rules $(\bigcirc a \to \bigcirc b) \in (12)^1$ and $(\neg b \to \bigcirc a) \in (13)^1$ so that, given T_0 we obtain $\bigcirc a \wedge \bigcirc b$, that is, $T_1 = \{a, b\}$. For $i = 2$, the involved rules are $(\bigcirc^2 a \to \bigcirc^2 b) \in (12)^2$ and $(\neg\bigcirc b \to \bigcirc^2 a) \in (13)^2$ so that, given T_1 we obtain $T_2 = \emptyset$. In a similar way, for $i = 3$ we have rules $(\bigcirc^3 a \to \bigcirc^3 b)$ and $(\neg\bigcirc^2 b \to \bigcirc^3 a)$ leading to $T_3 = \{a, b\}$ and then this behaviour is repeated. To sum up, we get a unique temporal equilibrium model $\langle \mathbf{T}, \mathbf{T}\rangle$ for Π_1 where $\mathbf{T}$ can be captured by the regular expression $(\ \emptyset\ \{a, b\}\ )^+$.

In some cases, however, we may face new situations that are not common in standard ASP. For instance, consider the formula (9), that corresponds to the conjunction of two temporal rules. As we saw before, this TLP has no temporal equilibrium models. Note that, by Theorem 18, this means that the ASP program $(9)^\omega$ has no stable models, although it is an *acyclic program* and (finite) acyclic programs always have a stable model. The intuitive reason for this is that atoms $\bigcirc^i p$ infinitely depend on the future, and there is no way to build an ordered proof starting from facts or the absence of them at a given end point.

5.2 *Reduction into TLP normal form*

The reduction into TLP normal form uses an extended signature $V_{\mathbf{L}}$ that contains an atom (a label) for each formula in the original language[1] $\mathcal{L}_V$, that is $V_{\mathbf{L}} = \{\mathbf{L}_\varphi \mid \varphi \in \mathcal{L}_V\}$. For convenience, we use $\mathbf{L}_\varphi \stackrel{\text{def}}{=} \varphi$ when φ is $\top$, $\bot$ or an atom $p \in V$. This allows us to consider $V_{\mathbf{L}}$ as a superset of V. For any non-atomic formula γ, its *definition*, $df(\gamma)$ corresponds to:

$$df(\gamma) \stackrel{\text{def}}{=} \begin{cases} \Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\varphi \bullet \mathbf{L}_\psi) & \text{if } \gamma = (\varphi \bullet \psi) \text{ with } \bullet \in \{\wedge, \vee, \rightarrow\}; \\ \Box(\mathbf{L}_\gamma \leftrightarrow \bigcirc \mathbf{L}_\varphi) & \text{if } \gamma = \bigcirc\varphi; \\ \Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\psi \vee (\mathbf{L}_\varphi \wedge \bigcirc \mathbf{L}_\gamma)\ ) & \\ \quad \wedge \Box(\mathbf{L}_\gamma \rightarrow \Diamond \mathbf{L}_\psi) & \text{if } \gamma = (\varphi\ \mathcal{U}\ \psi); \\ \Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\psi \wedge (\mathbf{L}_\varphi \vee \bigcirc \mathbf{L}_\gamma)\ ) & \\ \quad \wedge \Box(\Box \mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma) & \text{if } \gamma = (\varphi\ \mathcal{R}\ \psi). \end{cases}$$

Definition 19. *For any theory Γ in $\mathcal{L}_V$, we define the translation $\sigma(\Gamma)$ as:*

$$\sigma(\Gamma) \stackrel{\text{def}}{=} \{\mathbf{L}_\varphi \mid \varphi \in \Gamma\} \cup \{df(\gamma) \mid \gamma \in subf(\Gamma)\}$$

That is, $\sigma(\Gamma)$ collects the labels for all the formulas in Γ plus the definitions for all the subformulas in Γ. When the main connective in γ is a derived operator $\neg, \Diamond, \Box$, after simplifying truth constants, we obtain the following $df(\gamma)$:

$$df(\gamma) = \begin{cases} \Box(\mathbf{L}_\gamma \leftrightarrow \neg \mathbf{L}_\varphi\ ) & \text{if } \gamma = \neg\varphi; \\ \Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\varphi \vee \bigcirc \mathbf{L}_\gamma\ ) \wedge \Box(\mathbf{L}_\gamma \rightarrow \Diamond \mathbf{L}_\varphi) & \text{if } \gamma = \Diamond\varphi; \\ \Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\varphi \wedge \bigcirc \mathbf{L}_\gamma\ ) \wedge \Box(\Box \mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma) & \text{if } \gamma = \Box\varphi. \end{cases}$$

Let $\mathbf{T}$ be any LTL interpretation $\{I_i\}_{i \in \mathbb{N}}$ and V some set of atoms: we denote $\mathbf{T} \cap V$ to stand for the LTL interpretation $\{I'_i\}_{i \in \mathbb{N}}$ where $I'_i = I_i \cap V$. Similarly, given the THT interpretation $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$, we write $\mathbf{M} \cap V$ to stand for $\langle \mathbf{H} \cap V, \mathbf{T} \cap V \rangle$.

Lemma 20. *Let $\mathbf{M}$ be a model of a theory Γ in $\mathcal{L}_V$. Then, there exists some $\mathbf{M}'$ such that $\mathbf{M} = \mathbf{M}' \cap V$ and $\mathbf{M}' \models \sigma(\Gamma)$.*

Lemma 21. *Let Γ be a THT theory in $\mathcal{L}_V$ and $\mathbf{M}$ a model for $\sigma(\Gamma)$. Then for any $\gamma \in subf(\Gamma)$ and any $i \geq 0$, $\mathbf{M}(i, \mathbf{L}_\gamma) = \mathbf{M}(i, \gamma)$.*

Theorem 22. *For any theory Γ in $\mathcal{L}_V$: $\{\mathbf{M} \mid \mathbf{M} \models \Gamma\} = \{\mathbf{M}' \cap V \mid \mathbf{M}' \models \sigma(\Gamma)\}$.*

Clearly, the equality above is preserved if we include an arbitrary theory $\Gamma' \subseteq \mathcal{L}_V$ as follows $\{\mathbf{M} \mid \mathbf{M} \models \Gamma \cup \Gamma'\} = \{\mathbf{M}' \cap V \mid \mathbf{M}' \models \sigma(\Gamma) \cup \Gamma'\}$ and then we take the minimal models on both sides.

Since the transformation adds new auxiliary atoms, we must refine the idea of strong equivalence for theories with extended signatures.

Definition 23 (Strong faithfulness)**.** *We say that a translation $\sigma(\Gamma) \subseteq \mathcal{L}_U$ of some theory $\Gamma \subseteq \mathcal{L}_V$ with $V \subseteq U$ is* strongly faithful *if, for any theory $\Gamma' \subseteq \mathcal{L}_V$:*

$$Eq(V, \Gamma \cup \Gamma') = \{\mathbf{M} \cap V \mid \mathbf{M} \in Eq(U, \sigma(\Gamma) \cup \Gamma')\}$$

Corollary 24. *Translation $\sigma(\Gamma)$ is strongly faithful.*

[1] In this way, $V_{\mathbf{L}}$ is infinite, but when we later translate a given theory Γ, we can just take $V_{\mathbf{L}}$ as a label per each subformula.

Transformation $\sigma(\Gamma)$ is obviously modular, and its polynomial complexity can be easily deduced, but is not a temporal logic program yet, as it contains nested implications. However, we can apply some simple transformations on implication, conjunction and disjunction that have been shown to be strongly equivalent at the (non-temporal) propositional level[1] [9], and obtain a TLP without changing the signature $V_{\mathbf{L}}$. For each definition $df(\gamma)$, we define the strongly equivalent set (understood as the conjunction) of temporal logic program rules $df^*(\gamma)$ as shown in Figure 1. The temporal logic program $\sigma^*(\Gamma)$ is obtained by replacing in $\sigma(\Gamma)$ each subformula definition $df(\varphi)$ by the corresponding set of rules $df^*(\varphi)$. Note that, as $\sigma^*(\Gamma)$ is strongly equivalent to $\sigma(\Gamma)$ (under the same vocabulary) it preserves strong faithfulness with respect to Γ. Figure 2 shows the translation that results for derived operators after applying their definitions. To illustrate the effect of σ^* consider the example theory Γ_1 just consisting of $\Box(\neg p \rightarrow q \,\mathcal{U}\, p)$. The translation $\sigma^*(\Gamma_1)$ consists of the conjunction of $\mathbf{L}_4$ plus the rules in the $df^*(\gamma)$ columns of tables in Figure 3.

γ	$df(\gamma)$	$df^*(\gamma)$
$\varphi \wedge \psi$	$\Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\varphi \wedge \mathbf{L}_\psi)$	$\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\varphi)$ $\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\psi)$ $\Box(\mathbf{L}_\varphi \wedge \mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$
$\varphi \vee \psi$	$\Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\varphi \vee \mathbf{L}_\psi)$	$\Box(\mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\varphi \vee \mathbf{L}_\psi)$
$\varphi \rightarrow \psi$	$\Box(\mathbf{L}_\gamma \leftrightarrow (\mathbf{L}_\varphi \rightarrow \mathbf{L}_\psi)$	$\Box(\mathbf{L}_\gamma \wedge \mathbf{L}_\varphi \rightarrow \mathbf{L}_\psi)$ $\Box(\neg\mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\varphi \vee \neg\mathbf{L}_\psi \vee \mathbf{L}_\gamma)$
$\varphi \,\mathcal{U}\, \psi$	$\Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\psi \vee (\mathbf{L}_\varphi \wedge \bigcirc\mathbf{L}_\gamma))$ $\wedge\, \Box(\mathbf{L}_\gamma \rightarrow \Diamond\mathbf{L}_\psi)$	$\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\psi \vee \mathbf{L}_\varphi)$ $\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\psi \vee \bigcirc\mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\varphi \wedge \bigcirc\mathbf{L}_\gamma \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\gamma \rightarrow \Diamond\mathbf{L}_\psi)$
$\varphi \,\mathcal{R}\, \psi$	$\Box(\mathbf{L}_\gamma \leftrightarrow \mathbf{L}_\psi \wedge (\mathbf{L}_\varphi \vee \bigcirc\mathbf{L}_\gamma))$ $\wedge\, \Box(\Box\mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$	$\Box(\mathbf{L}_\psi \wedge \mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\psi \wedge \bigcirc\mathbf{L}_\gamma \rightarrow \mathbf{L}_\gamma)$ $\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\psi)$ $\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\varphi \vee \bigcirc\mathbf{L}_\gamma)$ $\Box(\Box\mathbf{L}_\psi \rightarrow \mathbf{L}_\gamma)$

Figure 1. Transformation $\sigma^*(\gamma)$ generating a temporal logic program.

Although $\sigma^*(\Gamma)$ is systematically applied on any subformula, for a practical implementation, we can frequently avoid the introduction of new labels, when the obtained expressions are already a TLP. For instance, in the example above, it would actually suffice with considering the replacement of $(p \,\mathcal{U}\, q)$ by label $\mathbf{L}_1$ to get $\Gamma_1 = \Box(\neg p \rightarrow \mathbf{L}_1)$ which is already a TLP. The next result shows that the computation of $\sigma^*(\Gamma)$ has a polynomial (in fact, linear) complexity on the size of Γ.

[1] These transformations for propositional operators contain expressions that are redundant in classical logic, but not in the logic of Here-and-There. The method in [10] can be used to show that these are, in fact, their possible minimal representations as sets of program rules.

γ	$df^*(\gamma)$	
$\neg\varphi$	$\Box(\mathbf{L}_\gamma \wedge \mathbf{L}_\varphi \rightarrow \bot)$	$\Box(\neg\mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$
$\Diamond\varphi$	$\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\varphi \vee \bigcirc\mathbf{L}_\gamma)$	$\Box(\mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$
	$\Box(\mathbf{L}_\gamma \rightarrow \Diamond\mathbf{L}_\varphi)$	$\Box(\bigcirc\mathbf{L}_\gamma \rightarrow \mathbf{L}_\gamma)$
$\Box\varphi$	$\Box(\mathbf{L}_\varphi \wedge \bigcirc\mathbf{L}_\gamma \rightarrow \mathbf{L}_\gamma)$	$\Box(\mathbf{L}_\gamma \rightarrow \mathbf{L}_\varphi)$
	$\Box(\Box\mathbf{L}_\varphi \rightarrow \mathbf{L}_\gamma)$	$\Box(\mathbf{L}_\gamma \rightarrow \bigcirc\mathbf{L}_\gamma)$

Figure 2. Transformation $\sigma^*(\gamma)$ that results for derived operators.

γ	$df^*(\gamma)$
$q\ \mathcal{U}\ p$	$\Box(\mathbf{L}_1 \rightarrow p \vee q)$
	$\Box(\mathbf{L}_1 \rightarrow p \vee \bigcirc\mathbf{L}_1)$
	$\Box(p \rightarrow \mathbf{L}_1)$
	$\Box(q \wedge \bigcirc\mathbf{L}_1 \rightarrow \mathbf{L}_1)$
	$\Box(\mathbf{L}_1 \rightarrow \Diamond p)$
$\neg p$	$\Box(\mathbf{L}_2 \wedge p \rightarrow \bot)$
	$\Box(\neg p \rightarrow \mathbf{L}_2)$

γ	$df^*(\gamma)$
$\neg p \rightarrow p\ \mathcal{U}\ q$	$\Box(\mathbf{L}_3 \wedge \mathbf{L}_2 \rightarrow \mathbf{L}_1)$
	$\Box(\neg\mathbf{L}_2 \rightarrow \mathbf{L}_3)$
	$\Box(\mathbf{L}_1 \rightarrow \mathbf{L}_3)$
	$\Box(\mathbf{L}_2 \vee \neg\mathbf{L}_1 \vee \mathbf{L}_3)$
$\Box(\neg p \rightarrow q\ \mathcal{U}\ p)$	$\Box(\mathbf{L}_3 \wedge \bigcirc\mathbf{L}_4 \rightarrow \mathbf{L}_4)$
	$\Box(\mathbf{L}_4 \rightarrow \mathbf{L}_3)$
	$\Box(\mathbf{L}_4 \rightarrow \bigcirc\mathbf{L}_4)$
	$\Box(\Box\mathbf{L}_3 \rightarrow \mathbf{L}_4)$

Figure 3. Transformation $\sigma^*(\Gamma_1)$ for example theory $\Gamma_1 = \{\Box(\neg p \rightarrow q\ \mathcal{U}\ p)\}$.

Theorem 25. *Translation $\sigma^*(\Gamma)$ is linear and its size can be bounded as follows: $size(\sigma^*(\Gamma)) \leq 2\ |\Gamma| + 34\ size(\Gamma)$.*

6. Computing Temporal Stable Models

In this section we consider two methods for computing temporal equilibrium/stable models. It must be noted that a temporal stable model has the form of an LTL-interpretation, that is, an infinite sequence of propositional interpretations. Furthermore, it is quite usual that a theory has an infinite set of temporal stable models. Thus, the usual behaviour of ASP solvers enumerating finite stable models is not applicable here. Fortunately, this infinite set of infinite structures is not arbitrary: temporal stable models show a regularity and, in fact, it can always be captured as the input language of a Büchi automaton[1] [4].

The two methods presented here are of different nature. The first method is more efficient and has been implemented in a tool that allows using variables, something more comfortable for practical examples. However, it is only applied to a syntactic subset of the normal form presented in the previous section, so it does not cover the full expressiveness of TEL. The second method accepts any arbitrary TEL theory and is based on an automata construction method. It has also helped to fix some complexity bounds for THT and TEL satisfiability. However, its current practical implementation generates automata of a rather large size, making this choice much more inefficient.

6.1 *Splitting and Loop Formulas*

We have seen that programs like (9) may have no temporal stable models due to, informally speaking, an "infinite dependence on the future." Fortunately, most ASP

[1] A Büchi automaton is a regular automaton for words of infinite length so that the word is accepted iff it visits an acceptance state an infinite number of times.

programs dealing with transition systems represent rules so that past does not depend on the future. This is what we called *future projected dependence* and can be captured by the following subclass of TLPs.

Definition 26 (Splitable TLP (STLP)). *A TLP* Π *for signature At is said to be* splitable *if* Π *consists of rules of any of the forms:*

$$B \wedge N \rightarrow H \tag{14}$$

$$B \wedge \bigcirc B' \wedge N \wedge \bigcirc N' \rightarrow \bigcirc H' \tag{15}$$

$$\Box(B \wedge \bigcirc B' \wedge N \wedge \bigcirc N' \rightarrow \bigcirc H') \tag{16}$$

where B *and* B' *are conjunctions of atoms,* N *and* N' *are conjunctions of negative literals like* $\neg p$ *with* $p \in At$*, and* H *and* H' *are disjunctions of atoms.* ⊠

The set of rules of form (14) in Π will be denoted $ini_0(\Pi)$ and correspond to initial rules for situation 0. The rules of form (15) in Π will be represented as $ini_1(\Pi)$ and are initial rules for the transition between situations 0 and 1. Finally, the set of rules of form (16) is written $dyn(\Pi)$ and contains dynamic rules. Both in (15) and (16), we understand that operator $\bigcirc$ is actually shifted until it only affects to atoms – this is always possible due to equivalences (3), (4). By abuse of notation, we will also use the formulas B, B', N, N', H and H' as sets that respectively contain the atoms that occur in each respective formula.

Notice that a rule of the form $\Box(B \wedge N \rightarrow H)$ (i.e., without $\bigcirc$ operator) is not splitable but can be transformed into the equivalent pair of rules $B \wedge N \rightarrow H$ and $\Box(\bigcirc B \wedge \bigcirc N \rightarrow \bigcirc H)$ which are both splitable. For instance, (12) becomes the pair of rules:

$$a \rightarrow b \tag{17}$$

$$\Box(\bigcirc a \rightarrow \bigcirc b) \tag{18}$$

As an example, (9) is not splitable, whereas Π_1={(11),(13),(17),(18)} is splitable being $ini_0(\Pi_1)$={(17)}, $ini_1(\Pi_1)$={(11)} and $dyn(\Pi_1)$={(17),(13)}. In particular, in (11) we have the non-empty sets $B' = \{b\}$, $N = \{a\}$ and $H' = \{a\}$, whereas for (13) the sets are $N = \{b\}$, $H' = \{a\}$.

The most interesting feature of splitable TLPs is that we can apply the so-called *splitting* technique [29] to obtain their temporal equilibrium models in an incremental way. Let us briefly recall this technique for the case of ASP programs. Following [29] we define:

Definition 27 (Splitting set). *Let* Π *be an ASP program consisting of (non-temporal) rules like* (14)*. Then a set of atoms* U *is a* splitting set *for* Π *if, for any rule like* (14) *in* Π*: if* $H \cap U \neq \emptyset$ *then* $(B \cup N \cup H) \subseteq U$*. The set of rules satisfying* $(B \cup N \cup H) \subseteq U$ *are denoted as* $b_U(\Pi)$ *and called the* bottom *of* Π *with respect to* U*.* ⊠

Consider the program:

$$a \rightarrow c \tag{19}$$

$$b \rightarrow d \tag{20}$$

$$\neg b \rightarrow a \tag{21}$$

$$\neg a \rightarrow b \tag{22}$$

The set $U = \{a, b\}$ is a splitting set for Π being $b_U(\Pi) = \{(21), (22)\}$. The idea of splitting is that we can compute first each stable model X of $b_U(\Pi)$ and then use the truth values in X for simplifying the program $\Pi \setminus b_U(\Pi)$ from which the rest of truth values for atoms not in U can be obtained. Formally, given $X \subseteq U \subseteq At$ and an ASP program Π, for each rule r like (14) in Π such that $B \cap U \subseteq X$ and $N \cap U$ is disjoint from X, take the rule $r^\bullet : B^\bullet \wedge N^\bullet \rightarrow H$ where $B^\bullet = (B \setminus U)$ and $N^\bullet = (N \setminus U)$. The program consisting of all rules $r^\bullet$ obtained in this way is denoted as $e_U(\Pi, X)$. Note that this program is equivalent to replacing in all rules in Π each atom $p \in U$ by $\bot$ if $p \notin X$ and by $\top$ if $p \in X$.

In the previous example, the stable models of $b_U(\Pi)$ are $\{a\}$ and $\{b\}$. For the first stable model $X = \{a\}$, we get $e_U(\Pi \setminus b_U(\Pi), \{a\}) = \{\top \rightarrow c\}$ so that $X \cup \{c\} = \{a, c\}$ should be a stable model for the complete program Π. Similarly, for $X = \{b\}$ we get $e_U(\Pi \setminus b_U(\Pi), \{b\}) = \{\top \rightarrow d\}$ and a "completed" stable model $X \cup \{d\} = \{b, d\}$. The following result guarantees the correctness of this method in the general case.

Theorem 28 (from [29])**.** *Let U be a splitting set for a set of rules Π like (14). A set of atoms X is a stable model of Π if, and only if both*

(i) *$X \cap U$ is a stable model of $b_U(\Pi)$;*
(ii) *and $X \setminus U$ is a stable model of $e_U(\Pi \setminus b_U(\Pi), X \cap U)$.* ⊠

In [29] this result was generalised for an infinite sequence of splitting sets, showing an example of a logic program with variables and a function symbol, so that the ground program was infinite. We adapt next this splitting sequence result for the case of splitable TLPs in TEL.

From Definition 17 we can easily conclude that, when Π is a splitable TLP, its program expansions have the form $\Pi^0 = ini_0(\Pi)$ and $\Pi^i = ini_0(\Pi) \cup ini_1(\Pi) \cup dyn(\Pi)^i$ for $i > 0$.

Proposition 29. *Given a splitable TLP Π for signature At and any $i \geq 0$:*

(i) *At^i is a splitting set for Π^ω;*
(ii) *and $b_{At^i}(\Pi^\omega) = \Pi^i$.* ⊠

Given any rule like r like (15) of (16) and a set of atoms X, we define its *simplification* $simp(r, X)$ as:

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

Given some LTL interpretation $\mathbf{T}$, let us define now the sequence of programs:

$$\Pi[\mathbf{T}, i] \stackrel{\text{def}}{=} e_{At^i}\left(\Pi^\omega \setminus \Pi^i\,,\, \mathbf{T}^i\right)$$

that is, $\Pi[\mathbf{T}, i]$ is the "simplification" of Π^ω by replacing atoms in At^i by their truth value with respect to $\mathbf{T}^i$. Then, we have:

Proposition 30.

$$\Pi[\mathbf{T}, 0] = (dyn(\Pi)^\omega \setminus dyn(\Pi)^1) \cup \{simp(r, T_0) \mid r \in ini_1(\Pi) \cup dyn(\Pi)\}$$
$$\Pi[\mathbf{T}, i] = (dyn(\Pi)^\omega \setminus dyn(\Pi)^{i+1}) \cup \{\bigcirc^i simp(r, T_i) \mid r \in dyn(\Pi)\}$$

for any $i \geq 1$. ⊠

As we can see, programs $\Pi[\mathbf{T}, i]$ maintain most part of $dyn(\Pi)^\omega$ and only differ in

simplified rules. Let us call these sets of simplified rules:

$$slice(\Pi, \mathbf{T}, 0) \stackrel{\text{def}}{=} \Pi^0 = ini_0(\Pi)$$

$$slice(\Pi, \mathbf{T}, 1) \stackrel{\text{def}}{=} \{simp(r, T_0) \mid r \in ini_1(\Pi) \cup dyn(\Pi)\}$$

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

Theorem 31 (Splitting Sequence Theorem)**.** *Let $\langle \mathbf{T}, \mathbf{T} \rangle$ be a model of a splitable 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* $i \geq 2$. ⊠

As an example, let us take again program $\Pi_1 = (11), (17), (18), (13)$. The program $\Pi_1^0 = ini_0(\Pi_1) = (17)$ has the stable model $\mathbf{T}^0 = \emptyset = T_0$. Then we take $slice(\Pi, \mathbf{T}, 1) = \{simp(r, T_0) \mid r \in ini_1(\Pi) \cup dyn(\Pi)\}$ that corresponds to $\{(\bigcirc b \rightarrow \bigcirc a), (\bigcirc a \rightarrow \bigcirc b), (\top \rightarrow \bigcirc a)\}$ whose stable model is $\{\bigcirc a, \bigcirc b\} = (\mathbf{T}^1 \setminus At^0)$ so that $T_1 = \{a, b\}$. In the next step, $slice(\Pi, \mathbf{T}, 2) = \{\bigcirc simp(r, T_1) \mid r \in dyn(\Pi)\} = \{(\bigcirc^2 a \rightarrow \bigcirc^2 b), (\top)\}$ whose stable model is $\emptyset = (\mathbf{T}^2 \setminus At^1)$ so that $T_2 = \emptyset$. Then, we would go on with $slice(\Pi, \mathbf{T}, 3) = \{\bigcirc^2 simp(r, T_2) \mid r \in dyn(\Pi)\} = \{(\bigcirc^3 a \rightarrow \bigcirc^3 b), (\top \rightarrow \bigcirc^3 a)\}$ leading to $\{\bigcirc^3 a, \bigcirc^3 b\}$ that is $T_3 = \{a, b\}$ and so on.

Theorem 31 allows us building the temporal equilibrium models by considering an infinite sequence of finite ASP programs $slice(\Pi, \mathbf{T}, i)$. If we consider each program $\Pi' = slice(\Pi, \mathbf{T}, i+1)$ for signature $At^{i+1} \setminus At^i$ then, since it is a standard disjunctive ASP program, we can use the main result in [16] to compute its stable models by obtaining the classical models of a theory $\Pi' \cup LF(\Pi')$ where LF stands for *loop formulas*. To make the paper self-contained, we recall next some definitions and results from [16].

Given an ASP program Π we define its *(positive) dependency graph* $G(\Pi)$ where its vertices are At (the atoms in Π) and its edges are $E \subseteq At \times At$ so that $(p, p) \in E$ for any atom[1] p, and $(p, q) \in E$ if there is an ASP rule in Π like (14) with $p \in H$ and $q \in B$. A nonempty set L of atoms is called a *loop* of a program Π if, for every pair p, q of atoms in L, there exists a path from p to q in $G(\Pi)$ such that all vertices in the path belong to L. In other words, L is a loop of iff the subgraph of $G(\Pi)$ induced by L is strongly connected. Notice that reflexivity of $G(\Pi)$ implies that for any atom p, the singleton $\{p\}$ is also a loop.

Definition 32 (external support)**.** *Given an ASP program Π for signature At, the* external support *formula of a set of atoms $Y \subseteq At$ with respect to Π, written $ES_\Pi(Y)$ is defined by:*

$$\bigvee_{r \in R(Y)} \left(B \wedge N \wedge \bigwedge_{p \in H \setminus Y} \neg p \right)$$

where $R(Y) = \{r \in \Pi$ like $(14) \mid H \cap Y \neq \emptyset$ and $B \cap Y = \emptyset\}$. ⊠

Theorem 33 (from [16])**.** *Given a program Π for signature At, and a (classical) model $X \subseteq At$ of Π then X is a stable model of Π iff for every loop Y of Π, X satisfies $\bigvee_{p \in Y} p \rightarrow ES_\Pi(Y)$* ⊠

[1] The original formulation in [16] did not consider reflexive edges, dealing instead with the idea of paths of length 0.

This result can be directly applied to each finite ASP program $slice(\Pi, \mathbf{T}, i)$. As we have slice programs for $i = 0, 1, \ldots$, this means we would obtain an infinite sequence of classical theories (each program plus its loop formulas). Fortunately, these theories are not arbitrary. For situations $0, 1$, we may obtain loops induced by dependencies that are due to initial rules, but for $i \geq 2$ loops follow a *repetitive pattern*, so they can be easily captured using $\Box$ and $\bigcirc$ operators. Thus, we just need to consider loops for situations $0, 1, 2$ bearing in mind that any loop at level $i = 2$ will occur repeatedly from then on. Given a splitable TLP Π its associated dependency graph $G(\Pi)$ is generated from the expanded (ASP) program Π^2, so that its nodes are atoms in the signature At^2 and its loops are obtained from this finite program. For instance, given our example program Π_1, its graph $G(\Pi_1)$, shown in Figure 4, is obtained from the expanded program Π_1^2 and, as we can see, contains the loops $\{\bigcirc a, \bigcirc b\}$ plus $\{A\}$ for any $A \in At^2$.

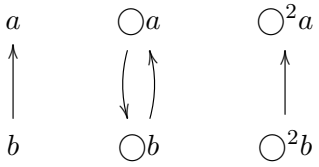

Figure 4. Graph $G(\Pi_1)$ (reflexive arcs are not displayed) corresponding to Π_1^2.

Theorem 34. *Let* Π *be a splitable TLP and* $\mathbf{T}$ *an LTL model of* Π*. Then* $\langle \mathbf{T}, \mathbf{T} \rangle$ *is a temporal equilibrium model of* Π *iff* $\mathbf{T}$ *is an LTL model of the union of formulas* $LF(Y)$ *defined as:*

$$\begin{array}{ll} Y^{\vee} \rightarrow ES_{ini_0(\Pi)}(Y) & \textit{for any loop } Y \subseteq At^0 = At \\ Y^{\vee} \rightarrow ES_{ini_1(\Pi) \cup dyn(\Pi)^1}(Y) & \textit{for any loop } Y \subseteq (At^1 \setminus At^0) \\ \Box\Big(Y^{\vee} \rightarrow ES_{dyn(\Pi)^2 \setminus dyn(\Pi)^1}(Y)\Big) & \textit{for any loop } Y \subseteq (At^2 \setminus At^1) \quad \boxtimes \end{array}$$

In our running example Π_1 we have $At^0 = \{a, b\}$ and $ini_0(\Pi) = (17)$ with two loops $\{a\}, \{b\}$ where $LF(\{a\}) = (a \rightarrow \bot)$ and $LF(\{b\}) = (b \rightarrow a)$. For $(At^1 \setminus At^0) = \{\bigcirc a, \bigcirc b\}$ we take the program $ini_1(\Pi_1) \cup dyn(\Pi)^1)$, that is, rules (11), (18), (13) ignoring $\Box$. We get three loops leading to the corresponding loop formulas $(\bigcirc a \rightarrow (\neg a \wedge \bigcirc b) \vee \neg b)$, $(\bigcirc b \rightarrow \bigcirc a)$ and $(\bigcirc a \vee \bigcirc b \rightarrow \neg b)$. Finally, for $At^2 \setminus At^1$ we have two loop formulas $\Box(\bigcirc^2 b \rightarrow \bigcirc^2 a)$ and $\Box(\bigcirc^2 a \rightarrow \neg \bigcirc b)$. It is not difficult to see that $\Pi_1 \cup LF(\Pi_1)$ is equivalent to the LTL theory: $\neg a \wedge \neg b \wedge \Box(\bigcirc a \leftrightarrow \neg b) \wedge \Box(\bigcirc b \leftrightarrow \neg b)$.

The technique of generating loop formulas has been implemented in a tool called `STeLP`[1] [7] that allows computing the temporal equilibrium models of an STLP. The input programs of `STeLP` adopt the standard ASP notation for conjunction, negation and implication, so that, an initial rule like (14) is represented as:

$$A_{m+1} \texttt{ v } \ldots \texttt{ v } A_s \texttt{ :- } A_1\texttt{, } \ldots \texttt{, } A_n\texttt{, not } A_{n+1}\texttt{, } \ldots \texttt{, not } A_m$$

Operator '$\bigcirc$' is represented as '`o`' whereas a dynamic rule like $\Box(\alpha \rightarrow \beta)$ is written as β `::-` α. Using this notation, program Π_1 becomes:

```
o a :- not a, o b.        b ::- a.        o a ::- not b.
```

[1] A `STeLP` web version is available at `http://kr.irlab.org/stelp`

Constraints in STeLP are more general than in STLP: their body can include any arbitrary combination of propositional connectives with `o`, `always` (standing for $\Box$) and `until` (standing for $\mathcal{U}$). The empty head $\bot$ is not represented. For instance, $\Box(\bigcirc a \wedge \neg b \rightarrow \bot)$ and $\Box\neg g \rightarrow \bot$ are constraints written as:

```
::- o a, not b.       :- always not g.
```

In `STeLP` we can also use rules where atoms have variable arguments like `p(`X_1`,...,`X_n`)` and, as happens with most ASP solvers, these are understood as abbreviations of all their ground instances. For more details on the input language, the reader is referred to [1]. As an example of `STeLP`domain, consider the classical puzzle where we have a wolf `w`, a sheep `s` and a cabbage `c` at one bank of a river. We have to cross the river carrying at most one object at a time. The wolf eats the sheep, and the sheep eats the cabbage, if nobody is around. Action `m(X)` means that we move some item `w,s,c` from one bank to the other. We assume that the boat is always switching between both banks of the river, so when no action is executed, this means we moved the boat without carrying anything. We will use a unique fluent `at(Y,B)` meaning that `Y` is at bank `B` being `Y` an item or the boat `b`. The complete encoding is shown in Figure 6.1.

```
% Domain predicates
domain item(X), object(Y).
static opp/2.     fluent at/2.      action m/1.
opp(l,r). opp(r,l).     item(w). item(s). item(c).
object(Z) :- item(Z).   object(b).
o at(X,A) ::- at(X,B), m(X), opp(A,B).        % Effect axiom for moving
o at(b,A) ::- at(b,B), opp(A,B).              % The boat is always moving
::- m(X), at(b,A), at(X,B), opp(A,B).         % Action executability
::- at(Y,A), at(Y,B), opp(A,B).               % Unique value constraint
o at(Y,A) ::- at(Y,A), not o at(Y,B),opp(A,B).% Inertia
::- at(w,A), at(s,A), at(b,B), opp(A,B).      % Wolf eats sheep
::- at(s,A), at(c,A), at(b,B), opp(A,B).      % Sheep eats cabbage
a(X) ::- not m(X).                            % Choice rules for action
m(X) ::- not a(X).                            %    execution
::- m(X), item(Z), m(Z), X != Z.              % Non-concurrent actions
at(Y,l).                                      % Initial state
g ::- at(w,r), at(s,r), at(c,r).              % Goal predicate
:- always not g.                              % Goal must be satisfied
```

Figure 5. Wolf-sheep-cabbage puzzle in `STeLP`.

The result obtained by `STeLP`for this example[1] is shown in Figure 6. As an example of non-existence of plan, if we further include the rule `::- at(w,r), at(c,r), at(b,l)` meaning that we cannot leave the wolf and the cabbage alone in the right bank, then the problem becomes unsolvable (we get a Büchi automaton with no accepting path).

6.2 *Automata based method*

In this section, following [6], we will provide an automata-based approach to determine whether a formula φ built over the propositional variables $\{p_1, \ldots, p_n\}$ has a TEL model. This method has the advantage of being applicable to any arbitrary temporal theory and also allows us to establish some complexity bounds for THT and TEL

[1]We removed the goal clauses for showing the whole system behaviour.

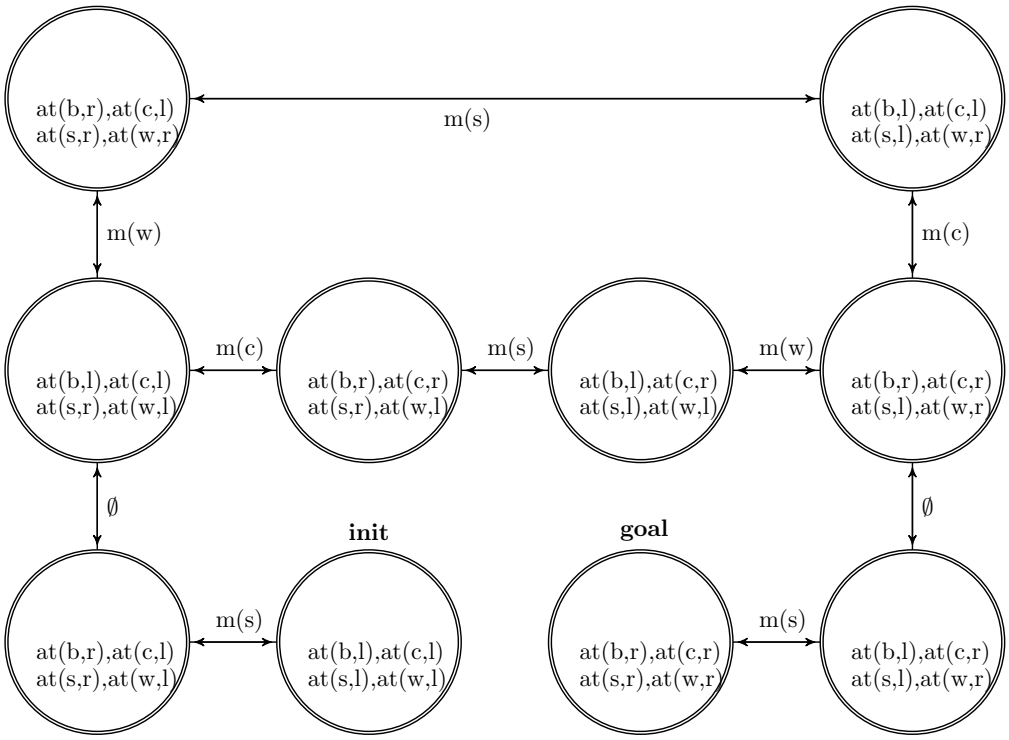

Figure 6. Automaton for the wolf-sheep-cabbage example.

satisfiability problems. The method builds a Büchi automaton $\mathcal{B}$ over the alphabet $\Sigma = \mathcal{P}(\{p_1, \ldots, p_n\})$ such that $\mathcal{L}(\mathcal{B})$ is equal to the set of TEL consequences for φ. Moreover, nonemptiness can be checked in ExpSpace, which allows to answer the open problem about the complexity of determining whether a temporal formula has a TEL model. This automata-based method has been implemented as a tool called `ABSTEM`[2].

For any LTL formula φ, there exist several algorithms [40] that allow obtaining a Büchi automaton that accepts the same ω-language than φ (see [6] for a recall of a Büchi automaton construction method). In a first step, we can just apply one of these algorithms to obtain a first automaton $\mathcal{A}_1$ that accepts LTL-models of φ or, equivalently, total THT-models of φ. Remember that temporal stable models are also LTL-models, so we need something else to reject those LTL-models that are not stable.

In a second step, we will strengthen the mapping φ' to obtain not only THT models of φ but also to constrain them to be strictly non-total (that is $\mathbf{H} < \mathbf{T}$) as follows

$$\varphi'' \stackrel{\text{def}}{=} \varphi' \wedge \bigvee_{i \in [1,n]} \Diamond((p_i' \rightarrow \bot) \wedge p_i)$$

φ'' characterizes the non-total THT models of the formula φ. The generalized disjunction ensures that at some position j, $H_j \subset T_j$ (strict inclusion).

Lemma 35. *The set of LTL models for the formula φ'' corresponds to the set of non-total THT models for the temporal formula φ.*

Let $\mathcal{A}_2$ be the Büchi automaton such that $\mathcal{L}(\mathcal{A}_2) = \text{Mod}(\varphi'')$, following again any construction similar to [40]. The set $\mathcal{L}(\mathcal{A}_2)$ is isomorphic to the set of non-total THT models of φ.

[2]Available at `http://kr.irlab.org/?q=abstem`

Given alphabet Σ we define the extended alphabet:

$$\Sigma' = \mathcal{P}(\{p_1, \ldots, p_n, p'_1, \ldots, p'_n\})$$

Let $h : \Sigma' \to \Sigma$ be a map (renaming) between the two finite alphabets such that $h(a) = a \cap \{p_1, \ldots, p_n\}$. h can be naturally extended as an homomorphism between finite words, infinite words and as a map between languages. Similary, given a Büchi automaton $\mathcal{A}_2 = (\Sigma', Q, Q_0, \delta, F)$, we write $h(\mathcal{A}_2)$ to denote the Büchi automaton $(\Sigma, Q, Q_0, \delta', F)$ such that $q \xrightarrow{a} q' \in \delta'$ iff there is $b \in \Sigma'$ such that $q \xrightarrow{b} q' \in \delta$ and $h(b) = a$. Obviously, $\mathcal{L}(h(\mathcal{A}_2)) = h(\mathcal{L}(\mathcal{A}_2))$ and $\mathcal{L}(h(\mathcal{A}_2))$ can be viewed as the set of total THT models for φ having a strictly smaller THT model.

The only remaining task is getting the intersection of total models captured by $\mathcal{L}(\mathcal{A}_1)$ with models of the negation of $\mathcal{L}(h(\mathcal{A}_2))$, that is, those for which we do not have a strictly smaller φ-model. The class of languages recognized by Büchi automata (the class of ω-regular languages) is effectively closed under union, intersection and complementation. Moreover, it is obviously closed under the renaming operation. Since $\mathcal{A}_1$, $\mathcal{A}_2$, and $h(\mathcal{A}_2)$ are Büchi automata, one can build a Büchi automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \Sigma^\omega \setminus \mathcal{L}(h(\mathcal{A}_2))$. Similarly, one can effectively build a Büchi automaton $\mathcal{B}_\varphi$ such that $\mathcal{L}(\mathcal{B}_\varphi) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}')$. Complementation can be performed using the constructions in [38] or in [37] (if optimality is required). Roughly speaking, complementation induces an exponential blow-up. As a result, we get the next proposition.

Proposition 36 (from [6])**.** *φ has a TEL model iff $\mathcal{L}(\mathcal{A}_1) \cap (\Sigma^\omega \setminus \mathcal{L}(h(\mathcal{A}_2))) \neq \emptyset$.*

Consequently, the set of TEL models for a given formula forms an ω-regular language.

Proposition 37. *For each temporal formula φ, one can effectively build a Büchi automaton that accepts exactly the TEL models for φ.*

The previous results were used in [6] to prove the following complexity bounds.

Proposition 38. *Checking whether a TEL formula has a TEL model can be done in* ExpSpace*.*

Theorem 39. *Checking whether a TEL formula has a TEL model is* PSpace*-hard.*

Theorem 40. *Checking whether two temporal formulae have the same TEL models is decidable in* ExpSpace *and it is* PSpace*-hard.*

7. Conclusions

We have provided a detailed survey on the main definitions and results for the hybrid formalism of Temporal Equilibrium Logic (TEL), an combination of Equilibrium Logic (a logical characterisation of answer set programming, ASP) with Linear-Time Temporal Logic.

Although some first relevant achievements have been obtained both in the theoretical study and its corresponding computation methods, there are however, many open topics and questions to be answered. At the theoretical level, for instance, the formal relation (Proposition 11) established between TEL and regular Equilibrium Logic (or ASP) is somehow weak. For instance, in the case of LTL, there exists a well-known correspondence [24] with FOL($<$) (First-order theory of linear order) – one may wonder whether this correspondence (or a similar one) is applicable for TEL and a Quantified Equilibrium Logic formulation of linear ordering.

Another missing result is the other direction for Theorem 14. This theorem allows us

fixing a sufficient condition for strong equivalence, but we ignore whether this condition is also necessary yet. This means in practice that when the current strong equivalence test provides a negative answer, we cannot guarantee that the compared theories are *not* strongly equivalent. In the non-temporal case, validity in the logic of Here-and-There has been proved to be a necessary and sufficient condition for strong equivalence. Existing tools for that case allow not only to answer when the strong equivalence is not satisfied, but even to provide a piece of logic program so that, when added to the compared theories, make them behave in a different way. A similar feature would be desirable for the case of TEL.

Finally, one more open issue is filling the complexity gap for TEL satisfiability. At the moment we know that this task falls between PSPACE and EXPSPACE.

Acknowledgements

This work is a byproduct of David Pearce's scientific program around Equilibrium Logic, a formalism introduced by him in 1996 that surprisingly connects intermediate logic and logic programming under the stable models semantics in a simple and natural way. David's contribution is not only limited to his personal achievements in the field, but also include his great ability to transmit the relevance of his work and the enthusiasm he puts on it to other researchers, having attracted a long list of collaborators. The combination of Equilibrium Logic with temporal logics was somehow an inevitable task that the authors began to explore five years ago, as part of a series of coordinated research projects under David's supervision.

References

[1] Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., Vidal, C.: Paving the way for temporal grounding. In: Proc. of the 28th International Conference on Logic Programming (ICLP'12). (2012)

[2] Aguado, F., Cabalar, P., Pérez, G., Vidal, C.: Strongly equivalent temporal logic programs. In: Proc. of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08). LNCS (vol. 5293). (2008) 8–20

[3] Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM **54**(12) (2011) 92–103

[4] Büchi, R.: On a decision method in restricted second-order arithmetic. In: Intl. Congress on Logic, Method and Philosophical Science'60. (1962) 1–11

[5] Cabalar, P.: A normal form for linear temporal equilibrium logic. In: Proc. of the 12th European Conference on Logics in Artificial Intelligence (JELIA'10). Lecture Notes in Computer Science. Volume 2258. Springer-Verlag (2010) 64–76

[6] Cabalar, P., Demri, S.: Automata-based computation of temporal equilibrium models. In: 21st International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'11). (2011)

[7] Cabalar, P., Diéguez, M.: `STeLP` – a tool for temporal answer set programming. In: Proc. of the 11th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'11). (2011)

[8] Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. Theory and Practice of Logic Programming **7**(6) (2007) 745–759

[9] Cabalar, P., Pearce, D., Valverde, A.: Reducing propositional theories in equilibrium logic to logic programs. In: Proc. of the 12th Portuguese Conference on Artificial Intelligence (EPIA'05). LNCS (3803), Springer-Verlag (2005) 4–17

[10] Cabalar, P., Pearce, D., Valverde, A.: Minimal logic programs. In: 23rd International Conference on Logic Programming (ICLP'07). LNCS (4670), Springer-Verlag (2007) 104–118
[11] Cabalar, P., Vega, G.P.: Temporal equilibrium logic: a first approach. In: Proc. of the 11th International Conference on Computer Aided Systems Theory, (EUROCAST'07). LNCS (4739). (2007) 241–248
[12] Calimeri, F., Ianni, G., Ricca, F., Alviano, M., Bria, A., Catalano, G., Cozza, S., Faber, W., Febbraro, O., Leone, N., Manna, M., Martello, A., Panetta, C., Perri, S., Reale, K., Santoro, M.C., Sirianni, M., Terracina, G., Veltri, P.: The third answer set programming competition: Preliminary report of the system competition track. In: Proc. of the 11th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'11). (2011) 388–403
[13] Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: The DLV$^{\mathrm{K}}$ planning system: Progress report. In: Proc. of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02). Volume 2424 of Lecture Notes in Computer Science. (2002) 541–544
[14] Fages, F.: Consistency of Clark's completion and existence of stable models. Methods of Logic in Computer Science **1** (1994) 51–60
[15] Ferraris, P.: Answer sets for propositional theories. In: Proc. of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05). LNCS (vol. 3662). (2005) 119–131
[16] Ferraris, P., Lee, J., Lifschitz, V.: A generalization of the Lin-Zhao theorem. Annals of Mathematics and Artificial Intelligence **47** (2006) 79–101
[17] Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07). (2007) 372–379
[18] Fisher, M.: A resolution method for temporal logic. In: Proceedings of the 12th international joint conference on Artificial Intelligence (IJCAI'91), Morgan Kaufmann Publishers Inc. (1991) 99–104
[19] Gebser, M., Grote, T., Schaub, T.: Coala: A compiler from action languages to ASP. In Janhunen, T., Niemelä, I., eds.: Proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA'10). Volume 6341 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2010) 360–364
[20] Gelfond, M., Inclezan, D.: Yet Another Modular Action Language. In: Proceedings of SEA-09, University of Bath Opus: Online Publications Store (2009) 64–78
[21] Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R.A., Bowen, K.A., eds.: Logic Programming: Proc. of the Fifth International Conference and Symposium (Volume 2). MIT Press, Cambridge, MA (1988) 1070–1080
[22] Gelfond, M., Lifschitz, V.: Action languages. Electronic Transactions on Artificial Intelligence. **2** (1998) 193–210
[23] Heyting, A.: Die formalen Regeln der intuitionistischen Logik. Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse (1930) 42–56
[24] Kamp, H.W.: Tense Logic and the Theory of Linear Order. Phd thesis, Computer Science Department, University of California at Los Angeles, USA (1968)
[25] Kautz, H.: The logic of persistence. In: Proc. of the 5th National Conference on Artificial Intelligence (AAAI'86). (1986) 401–405
[26] Kautz, H.A., Selman, B.: Planning as satisfiability. In: Proc. of the European Conf. on Artificial Intelligence (ECAI'92). (1992) 359–363
[27] Lifschitz, V.: Thirteen definitions of a stable model. In: Fields of Logic and Com-

putation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday. Volume 6300 of Lecture Notes in Computer Science., Springer (2010) 488–503
[28] Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. Computational Logic **2**(4) (2001) 526–541
[29] Lifschitz, V., Turner, H.: Splitting a logic program. In: Proceedings of the 11th International Conference on Logic programming (ICLP'94). (1994) 23–37
[30] Marek, V., Truszczyński, M. In: Stable models and an alternative logic programming paradigm. Springer-Verlag (1999) 169–181
[31] McCarthy, J.: Elaboration tolerance. In: Proc. of the 4th Symposium on Logical Formalizations of Commonsense Reasoning (Common Sense 98), London, UK (1998) 198–217
[32] McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence Journal **4** (1969) 463–512
[33] Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25** (1999) 241–273
[34] Pearce, D.: A new logical characterisation of stable models and answer sets. In: Proc. of Non monotonic extensions of logic programming (NMELP'96). (LNAI 1216). Springer-Verlag (1996)
[35] Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence **47**(1-2) (2006) 3–41
[36] Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press (1977) 46–57
[37] Safra, S.: Complexity of Automata on Infinite Objects. PhD thesis, The Weizmann Institute of Science, Rehovot (1989)
[38] Sistla, A., Vardi, M., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theoretical Computer Science **49** (1987) 217–237
[39] Turner, H.: Representing Actions in Logic Programs and Default Theories: A Situation Calculus Approach. Journal of Logic Programming **31 (1-3)** (1997) 245–298
[40] Vardi, M., Wolper, P.: Reasoning about infinite computations. Information & Computation **115** (1994) 1–37