A Logic for Signal Inserted Timed Frames

Jan Bergstra[§,†] Wan Fokkink[§] Kees Middelburg[§,‡,*]

[§] *Utrecht University, Department of Philosophy*
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

[†] *University of Amsterdam, Programming Research Group*
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

[‡] *KPN Research, Department of Network & Service Control*
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

janb@fwi.uva.nl *fokkink@phil.ruu.nl* *C.A.Middelburg@research.ptt.nl*

Abstract

We propose a first-order predicate logic TFL of timed frames extended with signals. This logic combines a simple syntax with a high expressivity; it can distinguish frames that are not the same as sets of transitions and states. We show how Dicky logic and CTL can be embedded in TFL.

1 Introduction

In recent years, a multitude of process algebras have evolved. Bergstra and Ponse [8, 9] proposed to study basic properties of such process algebras on the level of *frames*, which are labelled, directed graphs. Essentially, frames are transition systems without explicit start and termination nodes. Frames can be converted into processes by means of process extraction, which means that two states are singled out, which represent the start state and the successful termination state respectively. Thus, the algebra of frames constitutes a common platform for the study of basic properties of process algebras. In [7], frames have been extended with discrete time, following [4], and with signals, following [5]

Many process algebras have been supplied with various kinds of modal and temporal logics, usually with the aim to distinguish processes up to some desirable equivalence. That is, two processes are equivalent if and only if they make true exactly the same formulae in the logic. Examples of such an approach are Hennessy and Milner [18], where the logic distinguishes up to strong bisimulation,

[*] Currently on leave at International Institute for Software Technology, United Nations University, P.O. Box 3058, Macau, cam@iist.unu.edu.

and De Nicola and Vaandrager [14], where the logic distinguishes up to branching bisimulation.

In this paper, we propose a first-order predicate logic, called timed frame logic (TFL), for the algebra of signal-inserted timed frames, which may serve as a common platform for the diversity of process logics. We abandon the usual strategy for constructing logics in process theory. That is, we do not yet have a particular equivalence in mind. We aim for a logic which combines a simple syntax with a strong distinctive power. This is more conform the classical approach to modal logic; first, define a highly expressive logic, then determine sub-logics which distinguish up to some desirable relation. TFL distinguishes all frames that are not the same on a set theoretic level. TFL can serve as a basis for constructing sub-logics which distinguish up to nice equivalences such as strong bisimulation, trace equivalence, or σ-bisimulation from [7].

Unlike most other process logics, in TFL state names are allowed to occur in formulae explicitly. Owing to this particular feature, in TFL it is possible to express interesting properties such as 'the process contains a loop'. We show by a number of small examples, together with three larger examples based on specifications in SDL from [19], how basic properties of frames can be expressed in TFL.

Finally, we show how the modal logics Dicky logic [15] and CTL [12], which both make the theoretical basis for a model checker, can be translated into TFL. Hence, we obtain fragments of TFL where model checking is feasible.

Acknowledgements. Dennis Dams and Marco Hollenberg provided useful comments.

2 Timed Frames Logic

2.1 Timed frames

We define the notion of a *timed frame* [7]. We start from a countably infinite set $\mathbb{S}$ of state names $s, t, u, v, ...$ Furthermore, we assume a non-empty set A of actions $a, b, ...$, together with a special time step σ which represents evolving into the next time slice. In the sequel, μ ranges over $A \cup \{\sigma\}$, and this last set is abbreviated to A_σ.

A timed frame F is composed by means of the binary frame union $\oplus$ from atomic constructs of the following forms, where s, s' represent state names:

- s,
- $s \xrightarrow{a} s'$,
- $s \xrightarrow{\sigma} s'$.

The empty frame is denoted by $\emptyset$.

Table 1 presents the axioms FA1-6 for frames from [8] together with axioms TFA1,2 for timed frames from [7]. In the remainder of this section, $F = F'$ denotes that this equality between frames can be deduced from FA1-6+TFA1,2.

FA1	$X \oplus Y$	$=$	$Y \oplus X$
FA2	$(X \oplus Y) \oplus Z$	$=$	$X \oplus (Y \oplus Z)$
FA3	$X \oplus X$	$=$	X
FA4	$X \oplus \emptyset$	$=$	X
FA5	$s \oplus (s \xrightarrow{a} s')$	$=$	$s \xrightarrow{a} s'$
FA6	$s' \oplus (s \xrightarrow{a} s')$	$=$	$s \xrightarrow{a} s'$
TFA1	$s \oplus (s \xrightarrow{\sigma} s')$	$=$	$s \xrightarrow{\sigma} s'$
TFA2	$s' \oplus (s \xrightarrow{\sigma} s')$	$=$	$s \xrightarrow{\sigma} s'$

Table 1: Axioms for frames

Remark 2.1 A frame can be converted into a process using *process extraction*, from [9, 7], which means that a start state and a termination state in the frame are determined. The expression $s \overset{\frown}{\blacktriangleright} tF$ denotes the frame F with start state s and termination state t. See [7] for the precise definition of process extraction for timed frames.

2.2 Paths

We assume a set $Paths$ of *paths* $p, q, ...$, where $\mathbb{S} \subset Paths$, and a composition function $(\_, \_, \_) : Paths \times A_\sigma \times Paths \to Paths$. Paths are considered modulo associativity, that is,

$$(p_0, \mu_0, (p_1, \mu_1, p_2)) \;=\; ((p_0, \mu_0, p_1), \mu_1, p_2).$$

Also, we have the set $\mathbb{N}$ of natural numbers with the standard functions successor, addition, and multiplication. These functions are axiomatized by the standard equations.

The set $\pi \subseteq \mathbb{N} \times Paths \times \mathbb{S}$ contains the consecutive states of paths: $(n, p, s) \in \pi$ if and only if the nth state of p is s.

$$(0, s, s) \in \pi$$
$$(0, (s, \mu, p), s) \in \pi$$
$$(n+1, (s_0, \mu, p), s_1) \in \pi \text{ iff } (n, p, s_1) \in \pi.$$

Remark 2.2 A simpler version of π would be a function $\mathbb{N} \times Paths \to \mathbb{S}$, where $\pi(n, p)$ returns the nth state of p. However, a problem arises if p has length shorter than n; a special state 'bottom' is to be added in order to define $\pi(n, p)$ for such cases. The current formulation of π avoids the complication of adding such a bottom state.

The set $\alpha \subseteq \mathbb{N} \times Paths \times A_\sigma$ returns the consecutive actions of paths: $(n, p, \mu) \in \alpha$ if and only if the nth action of p is μ.

$$(0, (s, \mu, p), \mu) \in \alpha$$
$$(n+1, (s, \mu_0, p), \mu_1) \in \alpha \text{ iff } (n, p, \mu_1) \in \alpha.$$

Finally, we have a function E from frames to the power set of $Paths$, which for each frame F yields the paths that are induced by the frame F.

$s \in E(F)$ iff $F \oplus s = F$
$(s_0, \mu, s_1) \in E(F)$ iff $F \oplus (s_0 \xrightarrow{\mu} s_1) = F$
$(s_0, \mu_0, (s_1, \mu_1, p)) \in E(F)$ iff $F \oplus (s_0 \xrightarrow{\mu_0} s_1)) = F \ \wedge \ (s_1, \mu_1, p) \in E(F)$

2.3 The syntax

In this subsection we define the syntax of the timed frame logic TFL. We assume an initial model (see e.g. [10]) for $Paths$, which satisfies the required equations.

From now on, we assume four countably infinite sets $\mathcal{V}_S$ of variables, together with four sets $\mathbb{T}(S)$ of terms of sort S, for $S \in \{\mathbb{S}, A_\sigma, Paths, \mathbb{N}\}$. The sets of terms are defined inductively as follows.

- The collection $\mathbb{T}(\mathbb{S})$, with typical elements s, t, u, is $\mathbb{S} \cup \mathcal{V}_{\mathbb{S}}$.
- The collection $\mathbb{T}(A_\sigma)$, with typical element μ, is $A_\sigma \cup \mathcal{V}_{A_\sigma}$.
- The collection $\mathbb{T}(Paths)$, with typical elements p, q, is $Paths \cup \mathcal{V}_{Paths}$.
- The collection $\mathbb{T}(\mathbb{N})$, with typical elements m, n, is built from:
  - 0,
  - the variables in $\mathcal{V}_{\mathbb{N}}$,
  - $succ : \mathbb{T}(\mathbb{N}) \to \mathbb{T}(\mathbb{N})$,
  - the functions $+, \cdot : \mathbb{T}(\mathbb{N}) \times \mathbb{T}(\mathbb{N}) \to \mathbb{T}(\mathbb{N})$,

Let $\mu, \mu' \in \mathbb{T}(A_\sigma)$ and $s, s' \in \mathbb{T}(\mathbb{S})$ and $n, n' \in \mathbb{T}(\mathbb{N})$ and $p \in \mathbb{T}(Paths)$. The formulae ϕ, ψ in TFL are built inductively from the following constructs:

$$\left.\begin{array}{l} \mu = \mu' \\ s = s' \\ n = n' \\ \pi(n, p, s) \\ \alpha(n, p, \mu) \\ E(p) \end{array}\right\} \text{atomic formulae}$$

$$\left.\begin{array}{l} \phi \wedge \psi \\ \neg\phi \end{array}\right\} \text{boolean operators}$$

$$\left.\begin{array}{l} \forall \mu \in A_\sigma \ (\phi) \\ \forall s \in \mathbb{S} \ (\phi) \\ \forall p \in Paths \ (\phi) \\ \forall n \in \mathbb{N} \ (\phi) \end{array}\right\} \text{quantification}$$

We shall use the following abbreviations.

- $\phi \vee \psi$ for $\neg(\neg\phi \wedge \neg\psi)$,

- $\phi \Rightarrow \psi$ for $\neg\phi \vee \psi$,
- $n \leq n'$ for $\exists m \in \mathbb{N}\ (n + m = n')$,
- $s \xrightarrow{\mu} t$ for $E((s, \mu, t))$,
- $\exists\_ \in \_\ (\phi)$ for $\neg\forall\_ \in \_\ (\neg\phi)$.

2.4 The semantics

We define the semantics of TFL. $F \models_\rho \phi$ denotes that frame F makes true formula ϕ under the valuation ρ, which maps variables in $\mathcal{V}_S$ to closed terms in $\mathbb{T}(S)$, for $S \in \{\mathbb{S}, A_\sigma, \mathit{Paths}, \mathbb{N}\}$.

$$\begin{array}{rcl}
F \models_\rho a = a' & \Leftrightarrow & \rho(a) = \rho(a') \\
F \models_\rho s = s' & \Leftrightarrow & \rho(s) = \rho(s') \\
F \models_\rho n = n' & \Leftrightarrow & \rho(n) = \rho(n') \\
F \models_\rho \pi(n, p, s) & \Leftrightarrow & (\rho(n), \rho(p), \rho(s)) \in \pi \\
F \models_\rho \alpha(n, p, \mu) & \Leftrightarrow & (\rho(n), \rho(p), \rho(\mu)) \in \alpha \\
F \models_\rho E(p) & \Leftrightarrow & \rho(p) \in E(F) \\
F \models_\rho \phi \wedge \psi & \Leftrightarrow & F \models_\rho \phi \text{ and } F \models_\rho \psi \\
F \models_\rho \neg\phi & \Leftrightarrow & \text{not } F \models_\rho \phi \\
F \models_\rho \forall\mu \in A_\sigma\ (\phi) & \Leftrightarrow & \text{for all } \mu \in A_\sigma,\ F \models_{\rho(x \to a)} \phi \\
F \models_\rho \forall s \in \mathbb{S}\ (\phi) & \Leftrightarrow & \text{for all } s \in \mathbb{S},\ F \models_{\rho(x \to s)} \phi \\
F \models_\rho \forall p \in \mathit{Paths}\ (\phi) & \Leftrightarrow & \text{for all } p \in \mathit{Paths},\ F \models_{\rho(x \to p)} \phi \\
F \models_\rho \forall n \in \mathbb{N}\ (\phi) & \Leftrightarrow & \text{for all } n \in \mathbb{N},\ F \models_{\rho(x \to n)} \phi
\end{array}$$

In the sequel we write $F \models \phi$ for $\forall\rho(F \models_\rho \phi)$.

2.5 TFL distinguishes up to equality

The following simple example learns that TFL can distinguish frames which after process extraction (see [7]) yield the same term.

Example 2.3 Let $s \in \mathbb{S}$, and define $F_1 = \{s\}$ and $F_2 = \emptyset$. Process extraction with respect to F_1 and F_2 will yield immediate deadlock. However, $F_1 \models s$ and $F_2 \not\models s$.

TFL can distinguish timed frames up to their axiomatization FA1-6+TFA1,2 in Table 1.

Theorem 2.4 $\forall\phi\ (F_1 \models \phi \Leftrightarrow F_2 \models \phi) \iff \text{FA1-6+TFA1,2} \vdash F_1 = F_2$.

Proof. ($\Leftarrow$) Assume that FA1-6+TFA1,2 $\vdash F_1 = F_2$. Then it is not hard to see, by induction on the structure of frames, that for all $p \in \mathit{Paths}$, $p \in E(F_1)$ if and only if $p \in E(F_2)$. Then, by induction on the structure of formulae ϕ, it follows that $F_1 \models \phi$ if and only if $F_2 \models \phi$.

($\Rightarrow$) Assume that FA1-6+TFA1,2 $\not\vdash F_1 = F_2$. Then it is not hard to see that

1. either there is a state s such that $F_1 \oplus s = F_1$ is provable from the axioms, while $F_2 \oplus s = F_2$ is not provable from the axioms (or vice versa),

2. or there is a transition $s \xrightarrow{\mu} s'$ such that $F_1 \oplus (s \xrightarrow{\mu} s') = F_1$ is provable from the axioms, while $F_2 \oplus (s \xrightarrow{\mu} s') = F_2$ is not provable from the axioms (or vice versa).

In the first case, $F_1 \models s$ but $F_2 \not\models s$. In the second case, Then $F_1 \models s \xrightarrow{\mu} s'$ but $F_2 \not\models s \xrightarrow{\mu} s'$. $\Box$

Remark 2.5 For many purposes, the distinctive level of TFL is too fine. Future work will be to determine sub-logics of TFL which distinguish processes up to coarser process equivalences, such as strong bisimulation and trace equivalence. An interesting equivalence in this respect is the notion of σ-bisimulation, as defined in [7], which takes into account the special character of the time step σ. Notably, σ-bisimulation incorporates time non-determinism, that is, it does not distinguish whether a process p can evolve into both p' and p'' by the execution of a σ, or whether p can evolve into the alternative composition of p' and p'' by the execution of a σ.

2.6 Some expressions in TFL

We give some examples of expressions in TFL. For a start, we state some general properties of frames.

1. Each path has a start state:

$$\forall p \in \mathit{Paths}\ \exists s \in \mathbb{S}\ (\pi(0, p, s)).$$

2. Each path attains one state at a time:

$$\forall p \in \mathit{Paths}\ \forall s, t \in \mathbb{S}\ \forall n \in \mathbb{N}\ ((\pi(n, p, s) \wedge \pi(n, p, t)) \ \Rightarrow\ s = t).$$

Next, we assume a frame F, and we express some properties in TFL that may be valid for the frame F.

4. All paths in F that start in s once enter a loop:

$$\begin{array}{l}\exists n \in \mathbb{N}\ \forall p \in \mathit{Paths}\ (E(p)\ \wedge\ \pi(0, p, s)\\ \Rightarrow\ \exists t \in \mathbb{S}\ \exists k, l \in \mathbb{N}\ (k < l\ \wedge\ \pi(k, p, t)\ \wedge\ \pi(l, p, t))).\end{array}$$

5. Paths p in F that start in s and do progress beyond s' satisfy formula ϕ:

$$\begin{array}{l}\forall p \in \mathit{Paths}\ (E(p)\ \wedge\ \pi(0, p, s)\ \wedge\ \exists n \in \mathbb{N}\ \exists t \in \mathbb{S}\\ (\pi(n, p, s') \wedge \pi(n+1, p, t))\ \Rightarrow\ \phi).\end{array}$$

6. The process $s \curvearrowright F$ contains a deadlock:

$$\begin{array}{ll} & \exists u \in \mathbb{S}\ \exists p \in \mathit{Paths}\ \exists n \in \mathbb{N}\\ (& E(p)\\ \wedge & \pi(0, p, s)\\ \wedge & \pi(n, p, u)\\ \wedge & \forall m \in \mathbb{N}\ (1 \leq m \leq n\ \Rightarrow\ \neg\pi(m, p, t))\\ \wedge & \forall \mu \in A_\sigma\ \forall v \in \mathbb{S}\ (\neg(u \xrightarrow{\mu} v))\\ ). & \end{array}$$

2.7 Signals

In practical cases, behaviour can often only reach a specific situation if certain external conditions are satisfied. For example, a camp-fire can only be lit if it is not raining, or a train can only traverse a level crossing if its barriers are closed.

In order to capture such external conditions, Baeten and Bergstra [6, 3] extended the process algebra ACP with signals. A new approach towards signals was advocated in [5], and this approach was adapted to frames by Bergstra and Ponse [9]. Basically, a signal is a propositional formula, built from a collection of atomic propositions and $\mathbf{t}$ and $\mathbf{f}$ together with the connectives negation $\neg$ and conjunction $\wedge$. Each state can be supplied with a signal, which means that this state is accessible only if the signal holds. The construct $\Phi \leadsto F$ expresses that signal Φ is assigned to frame F, which means that the states s in $|F|$ are accessible only if Φ is true. For example, $(\mathbf{f} \leadsto s) \oplus (s \xrightarrow{\mu} s')$ equals $(\mathbf{f} \leadsto s) \oplus s'$, while $(\mathbf{t} \leadsto s) \oplus (s \xrightarrow{\mu} s')$ equals $s \xrightarrow{\mu} s'$.

Table 2 presents the axioms Ins1-8 for signal inserted frames from [9], together with axioms TIns1,2 from [7] to describe the interplay between signals and time.

Ins1	$\Phi \leadsto \emptyset$	$=$	$\emptyset$
Ins2	$\mathbf{t} \leadsto X$	$=$	X
Ins3	$\Phi \leadsto (\Psi \leadsto X)$	$=$	$(\Phi \wedge \Psi) \leadsto X$
Ins4	$(\Phi \leadsto X) \oplus (\Psi \leadsto X)$	$=$	$(\Phi \wedge \Psi) \leadsto X$
Ins5	$\Phi \leadsto (X \oplus Y)$	$=$	$(\Phi \leadsto X) \oplus (\Phi \leadsto Y)$
Ins6	$\Phi \leadsto (s \xrightarrow{a} s')$	$=$	$(\Phi \leadsto s) \oplus (s \xrightarrow{a} s') \oplus (\Phi \leadsto s')$
Ins7	$(\mathbf{f} \leadsto s) \oplus (s \xrightarrow{a} s')$	$=$	$(\mathbf{f} \leadsto s) \oplus s'$
Ins8	$(s \xrightarrow{a} s') \oplus (\mathbf{f} \leadsto s')$	$=$	$s \oplus (\mathbf{f} \leadsto s')$
TIns1	$\Phi \leadsto (s \xrightarrow{\sigma} s')$	$=$	$(\Phi \leadsto s) \oplus (s \xrightarrow{\sigma} s') \oplus (\Phi \leadsto s')$
TIns2	$(\Phi \leadsto s) \oplus (s \xrightarrow{\sigma} s')$	$=$	$(s \xrightarrow{\sigma} s') \oplus (\Phi \leadsto s')$

Table 2: Axioms for signal inserted frames

A complication in the synthesis of signals and time is that it is desirable that signals are not influenced by the passing of time, that is, if $s \xrightarrow{\sigma} s'$, then s' inherits the signal of s, and vice versa, see [7]. This interaction is expressed by axiom TIns2.

Table 3 presents axioms Ext1-6 from [7] which define a function $\chi(F, s)$, which extracts the signal that frame F induces on state s. Axiom Ext4 expresses that signals are inherited under time steps. Axiom Ext6 would not be sound if the condition $s' \oplus X \neq X$ were omitted from it. Namely, if s' occurs in X, then the signal of s' can be transmitted to s, if it is possible to evolve from s' into s by σ-transitions only. Hence, before Ext6 can be applied, first occurrences of s' in X have to be eliminated by means of axioms Ext2-4 and Ins3-6.

TFL is extended with signals by adding for each signal Φ and each state s an

Ext1	$\chi(\emptyset, s)$	$=$	$\mathbf{t}$
Ext2	$\chi(s' \oplus X, s)$	$=$	$\chi(X, s)$
Ext3	$\chi((s' \xrightarrow{a} s'') \oplus X, s)$	$=$	$\chi(X, s)$
Ext4	$\chi((s' \xrightarrow{\sigma} s'') \oplus X, s)$	$=$	$\chi((\chi(X, s'') \curvearrowright s') \oplus (\chi(X, s') \curvearrowright s'') \oplus X, s)$
Ext5	$\chi((\Phi \curvearrowright s) \oplus X, s)$	$=$	$\Phi \wedge \chi(X, s)$
Ext6	$\chi((\Phi \curvearrowright s') \oplus X, s)$	$=$	$\chi(X, s)$ if $s' \neq s$ and $s' \oplus X \neq X$

Table 3: Axioms for signal extraction

atomic formula $H(\Phi, s)$, which expresses intuitively that signal Φ holds in state s. We assume a complete proof system for propositional logic, see for example [9]. The extraction function χ is used in the semantics of TFL with signals, namely, we add the following clause to the semantics of TFL as defined in Section 2.4.

> $F \models_\rho H(\Phi, s) \;\Leftrightarrow\; (\chi(F, \rho(s)) \Rightarrow \Phi)$ can be derived from the axioms for signal inserted timed frames and for signal extraction together with the proof system for propositional logic.

Assume a signal inserted frame F. Perhaps surprisingly, the following TFL formula does *not* guarantee in general that state s' is not accessible from state s in F.

$$\exists p \in \mathit{Paths}\ \exists n \in \mathbb{N}\ (E(p) \wedge \pi(0, p, s) \wedge \pi(n, p, s')).$$

Namely, the path p may encounter a state t in between s and s' which is not accessible, because in F it is provided with the signal $\mathbf{f}$. In order to be sure that state s' is accessible from state s in F it is sufficient to validate the following formula.

$$\exists p \in \mathit{Paths}\ \exists n \in \mathbb{N}\ (E(p) \wedge \pi(0, p, s) \wedge \pi(n, p, s') \\ \wedge\ \forall m \leq n\ \forall t \in \mathbb{S}\ (\pi(m, p, t) \Rightarrow \neg H(\mathbf{f}, t))).$$

TFL extended with signals distinguishes signal inserted frames up to the axioms FA1-6+TFA1,2+Ins1-6+TIns1,2, under the assumption that signals Φ and Ψ are equivalent if $\Phi \Leftrightarrow \Psi$ can be derived from the proof system for propositional logic.

Theorem 2.6 $\forall \phi\ (F_1 \models \phi \Leftrightarrow F_2 \models \phi) \iff$ FA1-6+TFA1,2+Ins1-6+TIns1,2 $\vdash F_1 = F_2$.

Note that TFL does not respect the axioms Ins7,8. Namely, $(\mathbf{f} \curvearrowright s) \oplus (s \xrightarrow{a} s') \models s \xrightarrow{a} s'$, while $(\mathbf{f} \curvearrowright s) \oplus s' \not\models s \xrightarrow{a} s'$.

3 Some Examples

In order to give a feel of how TFL can be applied, we present three basic examples of signal-inserted timed frames, together with expressions in TFL which hold for these frames. These examples are based on specifications in Mauw [19],

where they are formulated in the language φSDL. Some of the properties in TFL that we will present are based on characteristics for these specifications that are described in [19]. In [19], information is exchanged between the thermometer and its environment. Here we abstract from this interaction, in order to keep the examples clean.

3.1 A thermometer

We consider a thermometer with a rough scale. If this thermometer is switched on, then it assumes that the temperature is undefined, and it performs a measurement. If this measurement does not agree with the assumption, or if they are both undefined, then the thermometer assumes the temperature it last measured, and after one time unit it performs a new measurement to corroborate the new assumption. This procedure is repeated until the measurement and the assumption agree, or in other words, until the thermometer measures the same temperature two times in a row. Then a beep signal is issued and the process ends.

We assume a data domain T of possible temperature values, which consists of integers in between -10 and 40 degrees Celcius, together with a value *undef*, which represents the temperatures that cannot be measured by the thermometer.

The state s_0 denotes the inactive thermometer. The state $s_1(k)$ for $k \in T$ represents the active thermometer, which assumes the temperature k.

The state s_0 can evolve either into the next time slice by the performance of a σ, or into $s_1(undef)$ by the performance of a start signal.

The state $s_1(k)$ for $k \in T$ can either perform a σ and evolve into a state $s_1(k')$ with $k' \in T$ unequal to k, or it can issue a beep and evolve into the inactive state s_0.

Summarizing, we have the actions σ and *start* and *beep*, and the following possible transitions:

$$\begin{array}{ll} & s_0 \xrightarrow{\sigma} s_0 \\ \oplus & s_0 \xrightarrow{start} s_1(undef) \\ \oplus & \bigoplus_{\{k,k' \in T | k \neq k' \vee k = undef\}} s_1(k) \xrightarrow{\sigma} s_1(k') \\ \oplus & \bigoplus_{\{k \in T | k \neq undef\}} s_1(k) \xrightarrow{beep} s_0 \end{array}$$

We formulate some requirements for this frame. The first two requirements stem from [19].

1. For each temperature k_0, the state $s_1(k_0)$ either performs a *beep*, or it evolves into a state $s_1(k)$ in the next time slice:

$$\begin{array}{l} \forall s' \in \mathbb{S}\ (s_1(k_0) \xrightarrow{a} s'\ \wedge\ \neg(a = beep)\ \Rightarrow \\ a = \sigma\ \wedge\ (s' = s_1(undef)\ \vee\ s' = s_1(-10)\ \vee \cdots \vee\ s' = s_1(40))). \end{array}$$

2. For each temperature $k_0 \neq undef$, the state $s_1(k_0)$ cannot evolve into itself in the next time slice:

$$\neg(s_1(k_0) \xrightarrow{\sigma} s_1(k_0)).$$

3. If a path starts in s_0, and at some time it performs a *beep*, then in the meantime it has evolved into a next time slice:

$$\forall p \in Paths\ \forall n \in \mathbb{N}\ (E(p)\ \wedge\ \pi(0,p,s_0)\ \wedge\ \alpha(n,p,beep)\ \Rightarrow \\ \exists m \in \mathbb{N}\ (m < n\ \wedge\ \alpha(m,p,\sigma))).$$

4. If a path starts in state s_0, and at some time it reaches a state $s_1(k_0)$, then in the meantime it has performed a *start*.

$$\forall p \in Paths\ \forall n \in \mathbb{N}\ (E(p)\ \wedge\ \pi(0,p,s_0)\ \wedge\ \pi(n,p,s_1(k_0))\ \Rightarrow \\ \exists m \in \mathbb{N}\ (m < n\ \wedge\ \alpha(m,p,start))).$$

3.2 An answering machine

We consider a simple telephone answering machine. When it perceives an incoming call, it waits for one time unit to see whether the receiver is lifted. If so, then the answering machine is de-activated. Otherwise, a pre-recorded tape is started, which contains some instructions, and subsequently the incoming call is recorded, for at most five time units. When the call is ended, or when the time is up, then the session is ended.

The state s_0 denotes the inactive answering machine. In state s_1, the answering machine has been activated. State s_2 represents the situation where the pre-recorded tape is played. In state $s_3(k)$ for $k = 0, ..., 5$, the incoming message is recorded, and there are at the most k time units to go before the session will be ended.

The state s_0 can evolve either into the next time slice by the performance of a σ, or into s_1 by the performance of a start signal.

The state s_1 can either return to s_0 if the incoming call is ended or if the receiver is lifted, or it can evolve into s_2 by the performance of a σ.

In s_2, the pre-recorded tape is played, after which it evolves into $s_3(5)$ by the performance of a σ.

The state $s_3(k)$ for $k > 0$ can either perform a σ and evolve into $s_3(k-1)$, or it can return to s_0 if the incoming call is ended. The state $s_3(0)$ ends the session and returns to s_0.

Summarizing, we have the actions σ and *start* and *end* and *receiver-lifted* and *call-ended*, and the following possible transitions:

$$\begin{array}{llllll} & s_0 \xrightarrow{\sigma} s_0 & \oplus & s_0 \xrightarrow{start} s_1 & & \\ \oplus & s_1 \xrightarrow{\sigma} s_2 & \oplus & s_1 \xrightarrow{receiver\text{-}lifted} s_0 & \oplus & s_1 \xrightarrow{call\text{-}ended} s_0 \\ \oplus & s_2 \xrightarrow{\sigma} s_3(5) & & & & \\ \bigoplus_{k=1}^{5} & s_3(k) \xrightarrow{\sigma} s_3(k-1) & \bigoplus_{k=1}^{5} & s_3(k) \xrightarrow{call\text{-}ended} s_0 & \oplus & s_3(0) \xrightarrow{end} s_0 \end{array}$$

We formulate two requirements for this frame, which originate from [19].

1. One session takes no more than eight time units.

$$\forall p \in Paths\ \forall n \in \mathbb{N}\ (E(p) \wedge \pi(0,p,s_0) \wedge \pi(n,p,s_0) \wedge \\ \forall m \in \mathbb{N}\ (0 < m < n \Rightarrow \neg\pi(m,p,s_0))\ \Rightarrow\ n \leq 9).$$

2. Each session is ended properly.

$$\forall p \in \mathit{Paths}\ \forall m, n \in \mathbb{N}\ (E(p) \wedge m < n \wedge \alpha(m, p, \mathit{start}) \wedge \alpha(n, p, \mathit{start}) \\ \Rightarrow\ \exists l \in \mathbb{N}\ (\alpha(l, p, \mathit{end}) \vee \alpha(l, p, \mathit{receiver\text{-}lifted}) \vee \alpha(l, p, \mathit{call\text{-}ended}))).$$

3.3 Traffic lights

We end this series of examples with a somewhat more complicated case study, which incorporates signals and communication. For a precise description how to deal with frame communication see [8].

We consider a crossing, regulated by two traffic lights A and B. In both directions, detectors are used to sense the presence of traffic. If a traffic light detects traffic, and if it is showing red, then it issues a request to be allowed to change to green. Then the other traffic light changes to red, possibly after some delay if it has been showing green only for a short period of time. Next, the first traffic light changes into green. The frame representation of the behaviour of the traffic light A is depicted in Figure 1. The signals *greenA* and *redA* express that traffic light A is showing green and red, respectively. The behaviour of traffic light B can be represented by the same frame, with the signals *greenB* and *redB* instead of *greenA* and *redA*, respectively.

The intuition for the frame in Figure 1 is as follows. Suppose that light A has just become green, which means that it is in the state 0, and that light B is showing red, which means that is in the state 5. Lights A and B pass through a σ-loop until B detects the presence of traffic, and it executes *detect*. Next, it emits a *request*, which communicates with *requested*. Thus, light B evolves into state 7, while light A evolves into state 3 either immediately, if in the meantime it has evolved into state 2, or after a delay of one time unit, otherwise. After a delay of one more time unit, in order to be sure that the crossing is empty, light A emits a *grant*, which communicates with *granted*, and lights A and B evolve into states 5 and 0, respectively.

The behaviour at the crossing consists of the merge of the two traffic lights, where one light is showing red and the other is showing green. That is, the full system is captured by the expression $\partial_{\{\mathit{request}, \mathit{requested}, \mathit{grant}, \mathit{granted}\}}(\mathit{red} \otimes \mathit{green})$. We formulate some requirements for this frame. The first two requirements originate from [19].

1. The lights are never green at the same time.

$$\forall p \in \mathit{Paths}\ ((E(p) \wedge \pi(0, p, \mathit{red} \otimes \mathit{green})) \Rightarrow \forall n \in \mathbb{N}\ (\neg\pi(n, p, \mathit{green} \otimes \mathit{green}))).$$

2. A detection of traffic is followed by a green light within two time units.

$$\forall p \in \mathit{Paths}\ ((E(p) \wedge \alpha(0, p, \mathit{detect})) \Rightarrow (\alpha(1, p, \mathit{request} \otimes \mathit{requested}) \\ \wedge\ (\alpha(3, p, \mathit{grant} \otimes \mathit{granted}) \vee \alpha(4, p, \mathit{grant} \otimes \mathit{granted})))).$$

3. If a traffic light turns into green, then traffic has been detected.

$$\forall p \in \mathit{Paths}\ \forall n \in \mathbb{N}\ ((E(p) \wedge \pi(0, p, \mathit{red} \otimes \mathit{green}) \wedge \pi(n, p, \mathit{green} \otimes \mathit{red})) \\ \Rightarrow \exists m < n\ (\alpha(m, p, \mathit{detect}))).$$

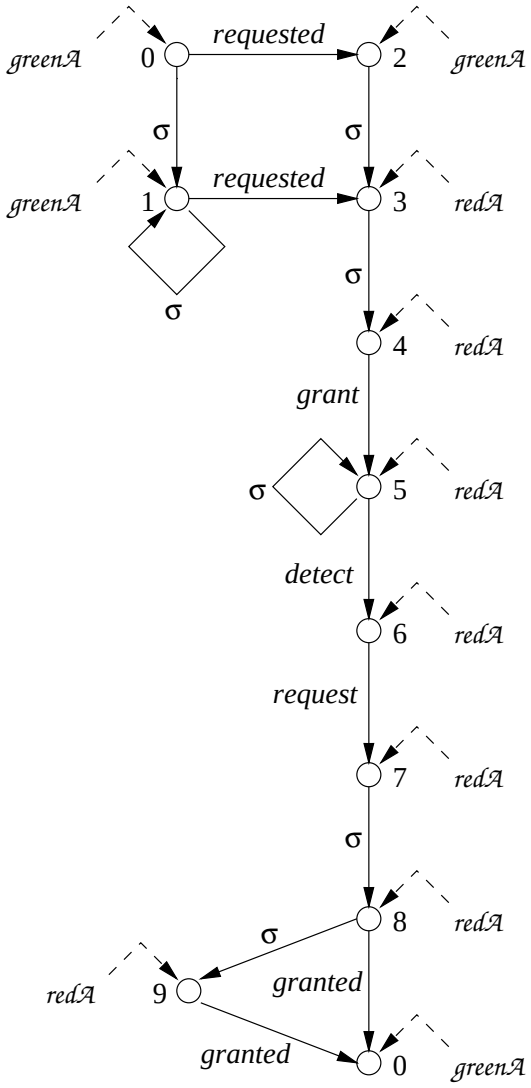

Figure 1: The behaviour of traffic light A

4 Comparison with Other Logics

We study how TFL relates to two other logics on finite transition systems in the literature, namely Dicky logic and computation tree logic.

4.1 Translation of Dicky logic into TFL

We study how the logic for finite transition systems from Dicky [15] relates to TFL. Dicky logic makes the theoretical basis for the tool Mec [1], which is tailored to computing boolean properties of finite transition systems.

We give a brief description of Dicky logic, following Arnold [2]. The logic assumes finite sets of states and of transition parameters. Dicky logic distinguishes between formulae on states and on transitions. First of all, there are sets of atomic propositions on states and on transitions. Now state and transition formulae are built inductively as follows:

- each atomic proposition on states is a state formula,
- each atomic proposition on transitions is a transition formula,
- if ϕ and ψ are state formulae, then $\phi \wedge_\sigma \psi$ and $\phi \vee_\sigma \psi$ and $\phi -_\sigma \psi$ are state formulae,
- if ϕ and ψ are transition formulae, then $\phi \wedge_\tau \psi$ and $\phi \vee_\tau \psi$ and $\phi -_\tau \psi$ are transition formulae,
- if ϕ is a transition formula, then $src(\phi)$ and $tgt(\phi)$ are state formulae,
- if ϕ is a state formula, then $in(\phi)$ and $out(\phi)$ are transition formulae.

State formulae ϕ are either valid or not valid in a state s, denoted by $s \models \phi$. Likewise, transition formulae ϕ are either valid or not valid with respect to a transition $s \xrightarrow{a} s'$, denoted by $s \xrightarrow{a} s' \models \phi$. The semantics of Dicky logic is defined as follows:

- Special functions assign to each state the atomic propositions on states that are valid in this state, and to each transition the atomic propositions on transitions that are valid with respect to this transition.
- $s \models \phi \wedge_\sigma \psi$ iff $s \models \phi$ and $s \models \psi$.
  $s \models \phi \vee_\sigma \psi$ iff $s \models \phi$ or $s \models \psi$.
  $s \models \phi -_\sigma \psi$ iff $s \models \phi$ and $s \not\models \psi$.
- $s \xrightarrow{a} s' \models \phi \wedge_\tau \psi$ iff $s \xrightarrow{a} s' \models \phi$ and $s \xrightarrow{a} s' \models \psi$.
  $s \xrightarrow{a} s' \models \phi \vee_\tau \psi$ iff $s \xrightarrow{a} s' \models \phi$ or $s \xrightarrow{a} s' \models \psi$.
  $s \xrightarrow{a} s' \models \phi -_\tau \psi$ iff $s \xrightarrow{a} s' \models \phi$ and $s \xrightarrow{a} s' \not\models \psi$.
- $s \models src(\phi)$ iff there is a transition $s \xrightarrow{a} s'$ such that $s \xrightarrow{a} s' \models \phi$.
  $s \models tgt(\phi)$ iff there is a transition $s' \xrightarrow{a} s$ such that $s' \xrightarrow{a} s \models \phi$.

- $s \xrightarrow{a} s' \models in(\phi)$ iff $s' \models \phi$.
  $s \xrightarrow{a} s' \models out(\phi)$ iff $s \models \phi$.

Assume non-empty, finite sets A of atoms and $\mathbb{S}$ of states. We assume atomic propositions Φ on states, together with a single atomic proposition $\Diamond$ on transitions. The intuition behind this last atomic proposition is that it yields the possible transitions. We define a translation T from expressions $s \models \phi$ and $s \xrightarrow{a} s' \models \phi$ in Dicky logic into TFL inductively as follows.

$$
\begin{array}{lcl}
T(s \models \Phi) & = & H(\Phi, s) \\
T(s \xrightarrow{a} s' \models \Diamond) & = & s \xrightarrow{a} s' \\
T(s \models \phi \wedge_\sigma \psi) & = & T(s \models \phi) \wedge T(s \models \psi) \\
T(s \models \phi \vee_\sigma \psi) & = & T(s \models \phi) \vee T(s \models \psi) \\
T(s \models \phi -_\sigma \psi) & = & T(s \models \phi) \wedge \neg T(s \models \psi) \\
T(s \xrightarrow{a} s' \models \phi \wedge_\tau \psi) & = & T(s \xrightarrow{a} s' \models \phi) \wedge T(s \xrightarrow{a} s' \models \psi) \\
T(s \xrightarrow{a} s' \models \phi \vee_\tau \psi) & = & T(s \xrightarrow{a} s' \models \phi) \vee T(s \xrightarrow{a} s' \models \psi) \\
T(s \xrightarrow{a} s' \models \phi -_\tau \psi) & = & T(s \xrightarrow{a} s' \models \phi) \wedge \neg T(s \xrightarrow{a} s' \models \psi) \\
T(s \models src(\phi)) & = & \exists a \in A\ \exists s \in \mathbb{S}\ (T(s \xrightarrow{a} s' \models \phi)) \\
T(s \models tgt(\phi)) & = & \exists a \in A\ \exists s \in \mathbb{S}\ (T(s' \xrightarrow{a} s \models \phi)) \\
T(s \xrightarrow{a} s' \models in(\phi)) & = & T(s' \models \phi) \\
T(s \xrightarrow{a} s' \models out(\phi) & = & T(s \models \phi)
\end{array}
$$

Let the frame F consist of the frame composition of the following expressions:

- $\Phi \frown s$ if the atomic proposition Φ holds in state s, or $(\neg\Phi) \frown s$ if Φ does not hold in s.
- $s \xrightarrow{a} s'$ if $s \xrightarrow{a} s' \models \Diamond$.

Then $s \models \phi$ holds in Dicky logic if and only if $F \models T(s \models \phi)$ holds in TFL, and $s \xrightarrow{a} s' \models \phi$ holds in Dicky logic if and only if $F \models T(s \xrightarrow{a} s' \models \phi)$ holds in TFL.

4.2 Translation of CTL into TFL

Computation tree logic (CTL) from Clarke and Emerson [11] is a popular formalism to express properties of transition systems. CTL allows model checking in

linear time [12], which has been automated in the tool EMC. We will determine a fragment of TFL that is equivalent with CTL, in order to have a sub-logic where model checking is feasible.

In a similar fashion we can establish a fragment of TFL that is equivalent with the logic CTL* [16], which is more expressive than CTL, because it handles quantification in a more liberal way. However, the model checking problem has been shown to be PSPACE-complete for CTL* [17, 20], so we refrain from this translation.

We give a brief description of CTL. This logic too assumes a finite set of states, and a transition system induced by a binary relation R between states. It is required that for each state s there is a state t with sRt. Hence, each path in the transition system can be extended to an infinite path in the transition system. Furthermore, CTL assumes a set of atomic propositions. Formulae in CTL are constructed inductively as follows, where ϕ and ψ represent formulae:

- each atomic proposition is a formula,

- $\neg\phi$ and $\phi \wedge \psi$ are formulae,

- $\forall\ [\phi\ U\ \psi]$ and $\exists\ [\phi\ U\ \psi]$ are formulae,

- $\exists\ X\ \phi$ is a formulae.

The expressions $\forall$ and $\exists$ denote universal and existential quantification over the collection of infinite paths in the transition system, respectively.

Formulae ϕ are either valid or not valid in a state s, denoted by $s \models \phi$. The semantics of CTL is defined as follows:

- A special function assigns to each state the atomic propositions that are valid in this state.

- $s \models \neg\phi$ iff $s \not\models \phi$.

  $s \models \phi \wedge \psi$ iff both $s \models \phi$ and $s \models \psi$.

- $s \models \forall\ [\phi\ U\ \psi]$ iff each infinite path that starts in s once reaches a state where ψ holds, and until that time ϕ holds along this path.

  $s \models \exists\ [\phi\ U\ \psi]$ iff there exists an infinite path that starts in s and once reaches a state where ψ holds, and until that time ϕ holds along this path.

- $s \models \exists\ X\ \phi$ iff there exists a state t such that sRt and ϕ holds in t.

Assume a finite set $\mathbb{S}$ of states. The set of atomic propositions contains elements Φ. We define a translation T from formulae $s \models \phi$ in CTL into TFL inductively

as follows.

$$
\begin{array}{lcl}
T(s \models \Phi) & = & H(\Phi, s) \\
T(s \models \neg\phi) & = & \neg T(s \models \phi) \\
T(s \models \phi \wedge \psi) & = & T(s \models \phi) \wedge T(s \models \psi) \\
T(s \models \forall\, [\phi\ U\ \psi]) & = & \forall q \in \mathit{Paths}\ (E(q) \wedge \pi(0, q, s)\ \Rightarrow \\
 & & \exists p \in \mathit{Paths}\ (E(p) \wedge \forall l \in \mathbb{N}\ \forall v \in \mathbb{S}\ (\pi(l, q, v) \\
 & & \Rightarrow \pi(l, p, v))\ \wedge\ \exists n \in \mathbb{N}\ \exists t \in \mathbb{S}\ (\pi(n, p, t) \wedge T(t \models \psi) \\
 & & \wedge\ \forall m < n\ \forall u \in \mathbb{S}\ (\pi(m, p, u) \Rightarrow T(u \models \phi))))) \\
T(s \models \exists\, [\phi\ U\ \psi]) & = & \exists p \in \mathit{Paths}\ (E(p)\ \wedge\ \pi(0, p, s) \wedge \\
 & & \exists n \in \mathbb{N}\ \exists t \in \mathbb{S}\ (\pi(n, p, t) \wedge T(t \models \psi) \wedge \\
 & & \forall m < n\ \forall u \in \mathbb{S}\ (\pi(m, p, u) \Rightarrow T(u \models \phi)))) \\
T(s \models \exists\ X\ \phi) & = & \exists t \in \mathbb{S}\ \exists p \in \mathit{Paths}\ (E(p) \wedge \pi(0, p, s) \wedge \pi(1, p, t) \\
 & & \wedge T(t \models \phi))
\end{array}
$$

The difficult case is the expression for $T(s \models \forall\, [\phi\ U\ \psi])$ in TFL, which says that each *infinite* path starting in s once reaches a state where ψ holds, and until that time ϕ holds along this infinite path. Since we consider finite paths, the TFL translation says: each (finite) path q starting in s can be extended to a (finite) path p such that p reaches a state t where ψ holds, and until that time ϕ holds along p.

Let the frame F consist of the frame composition of the following expressions:

- $\Phi \frown s$ if the atomic proposition Φ holds in state s, or $(\neg\Phi) \frown s$ if Φ does not hold in s.
- $s \longrightarrow s'$ if sRs'.

Then $s \models \phi$ holds in CTL if and only if $F \models T(s \models \phi)$ holds in TFL.

Remark 4.1 In spirit, TFL is even more closely related to ACTL* and ACTL from De Nicola and Vaandrager [13], which extends CTL* and CTL to labelled transition systems. Similar as we did for CTL, it is possible to give a direct translation of ACTL formulae into TFL. In [13] a linear algorithm is given to translate ACTL formulae into CTL formulae.

References

[1] A. Arnold. Mec: a system for constructing and analysing transition systems. In J. Sifakis, editor, *Proceedings Workshop on Automatic Verification Methods for Finite State Systems,* Grenoble, LNCS 407, pages 117–132. Springer, 1989.

[2] A. Arnold. *Finite Transition Systems.* Prentice Hall International, 1994.

[3] J.C.M. Baeten and J.A. Bergstra. Process algebra with signals and conditions. In M. Broy, editor, *Proceedings Summer School on Programming and Mathematical Methods,* Marktoberdorf, NATO ASI Series F88, pages 273–323. Springer, 1991.

[4] J.C.M. Baeten and J.A. Bergstra. Discrete time process algebra. Technical Report P9208c, University of Amsterdam, 1995. To appear in *Formal Aspects of Computing.*

[5] J.C.M. Baeten and J.A. Bergstra. Process algebra with propositional signals. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Proceedings 2nd Workshop on the Algebra of Communicating Processes (ACP'95),* Eindhoven, Report CS-95-14, pages 213–227. Eindhoven University of Technology, 1995.

[6] J.A. Bergstra. ACP with signals. In J. Grabowski, P. Lescanne, and W. Wechler, editors, *Algebraic and Logic Programming*, LNCS 343, pages 11–20. Springer, 1988.

[7] J.A. Bergstra, W.J. Fokkink, and C.A. Middelburg. Algebra of timed frames. Logic Group Preprint Series 148, Utrecht University, 1995.

[8] J.A. Bergstra and A. Ponse. Frame algebra with synchronous communication. In *Information Systems – Correctness and Reusablity*, pages 3–15. World Scientific, 1995.

[9] J.A. Bergstra and A. Ponse. Frame-based process logic. In *Workshop on Modal Logic and Process Algebra: a bisimulation perspective,* Amsterdam, *CSLI Lecture Notes* 53, pages 39–63. CSLI, Stanford University, 1995.

[10] C.C. Chang and H.J. Keisler. *Model Theory*, *Studies in Logic and the Foundations of Mathematics* 73. North-Holland, 1990.

[11] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Proceedings Workshop on Logics of Programs,* Yorktown Heights, NY, LNCS 131, pages 52–71. Springer, 1981.

[12] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 2(8):244–263, 1986.

[13] R. De Nicola and F.W. Vaandrager. Action versus state based logics for transition systems. In I. Guessarian, editor, *Proceedings LITP Spring School on Semantics of Systems of Concurrent Processes,* La Roche Posay, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer, 1990.

[14] R. De Nicola and F.W. Vaandrager. Three logics for branching bisimulation. *Journal of the ACM*, 42(2):458–487, 1995.

[15] A. Dicky. An algebraic and algorithmic method for analysing transition systems. *Theoretical Computer Science*, 46(2,3):285–303, 1986.

[16] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: on branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[17] E.A. Emerson and A.P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.

[18] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[19] S. Mauw. Example specifications in φSDL. Unpublished manuscript, 1995.

[20] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.