Positive and negative results on the decidability of the model-checking problem for an epistemic extension of Timed CTL

Cătălin Dima

LACL, Université Paris-Est Créteil Val-de-Marne

Séminaire MeFoSyLoMa, 6/05/2011

Epistemic logics

- Logics for reasoning about knowledge.
- Applications in artificial intelligence: multi-agent autonomous systems.
- Applications in economics: game-like situations.
- Applications in security: as an instance of multi-agent systems.
  - Expressing noninterference, anonymity, authentication, group properties.

Modal epistemic logics: syntax

- Atomic propositions: $Bob\_sent\_message\_M\_to\_Alice, Alice\_opened\_file\_f \ldots$
- Boolean connectives.
- Knowledge operators:
  - $K_{Alice}\phi$: *Alice* knows the truth value of ϕ.
  - $E_{Alice,Bob}\psi$: Both *Alice* and *Bob* know the truth value of ϕ ("everybody").
  - $C_{Alice,Bob}\psi$: Both *Alice* and *Bob* know the truth value of ϕ, and each knows that the other knows this, and each knows that the other knows that each knows this, and... ("common knowledge").

Specifying information flow properties in temporal epistemic logics

- Epistemic aspect of information flow properties.

$$\Diamond\Big(\neg K_{Alice} Bob\_opened\_file\_f \wedge \bigcirc K_{Alice} \bullet Bob\_opened\_file\_f\Big)$$

- [Dima & Enea '07]: syntactic and axiomatic presentation of information flow properties, based on *nondeducibility on strategies*.

Epistemic logics: semantics

- Kripke structure: states S, labeled with atomic propositions, $\nu : S \to 2^{\Pi}$.
- **Observability** (or **indistinguishability**) relation for each agent:

  s_1 : $Alice\_opened\_file\_f, Bob\_opened\_file\_f.$
  q_1 : $Alice\_opened\_file\_f, Bob\_opened\_file\_f, Bob\_opened\_file\_g.$
  s_2 : $Alice\_opened\_file\_f, Bob\_opened\_file\_f, Bob\_opened\_file\_h.$
  s_3 : $Alice\_opened\_file\_g, Bob\_opened\_file\_f.$
  q_2 : $Alice\_opened\_file\_g, Bob\_opened\_file\_g.$
- Alice observes color: $s_1 \sim_{Alice} q_1 \sim_{Alice} s_2 \not\sim_{Alice} s_3 \sim_{Alice} q_3$.
- Bob observes state names, but not indices: $s_1 \sim_{Bob} s_2 \sim_{Bob} s_3 \not\sim_{Bob} q_1 \sim_{Bob} q_2$.
- $M, s_1 \models K_{Alice}\ Bob\_opened\_file\_f.$
  - Alice, knowing system description, if observes green state, deduces $Bob\_opened\_file\_f.$
  - *Logical connection* (for *Alice*) between the green color and $Bob\_opened\_file\_f.$

Epistemic logics: semantics

- Kripke structure: states S, labeled with atomic propositions, $\nu : S \to 2^{\Pi}$.
- **Observability** (or **indistinguishability**) relation for each agent:

  s_1 : $Alice\_opened\_file\_f$, $Bob\_opened\_file\_f$.
  q_1 : $Alice\_opened\_file\_f$, $Bob\_opened\_file\_f$, $Bob\_opened\_file\_g$.
  s_2 : $Alice\_opened\_file\_f$, $Bob\_opened\_file\_f$, $Bob\_opened\_file\_h$.
  s_3 : $Alice\_opened\_file\_g$, $Bob\_opened\_file\_f$.
  q_2 : $Alice\_opened\_file\_g$, $Bob\_opened\_file\_g$.

- Alice observes color: $s_1 \sim_{Alice} q_1 \sim_{Alice} s_2 \not\sim_{Alice} s_3 \sim_{Alice} q_3$.
- Bob observes state names, but not indices: $s_1 \sim_{Bob} s_2 \sim_{Bob} s_3 \not\sim_{Bob} q_1 \sim_{Bob} q_2$.
- $M, s_1 \models K_{Alice}\, Bob\_opened\_file\_f$.
  - Alice, knowing system description, if observes green state, deduces $Bob\_opened\_file\_f$.
  - *Logical connection* (for *Alice*) between the green color and $Bob\_opened\_file\_f$.

Temporal epistemic logics

- Knowledge might be acquired in time, through observation of system evolution.
- Modal logics combining knowledge and temporal modalities:
  1. Enrichment of LTL ($\bigcirc\phi$, $\phi\mathcal{U}\psi$);
  2. Enrichment of CTL ($A\phi$ and dual $E\phi$);
  3. ATL ($\langle\langle Alice, Bob\rangle\rangle\phi$);
  4. Dynamic logics, μ-calculus with epistemic modalities.

CTLK, an epistemic variant of CTL

- CTL with knowledge modalities:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid A\bigcirc\phi \mid \phi A\mathcal{U}\phi \mid \phi E\mathcal{U}\phi \mid K_A\phi$$

  - $p \in \Pi$, set of atomic propositions.
  - $A \in Ag$, set of $n \geq 2$ agents.
- Dual operator: $P_A\phi = \neg K_A\neg\phi$.
- No common knowledge operator.

Semantics of temporal epistemic logics

- Transition system (for the temporal part).
  - (Instantaneous) states and transitions between states.
- Multi-agent Kripke structure (for the epistemic part).
  - Observability relations on states.
- Connections between the two structures.
  - System state = run.

Semantics of temporal epistemic logics (2)

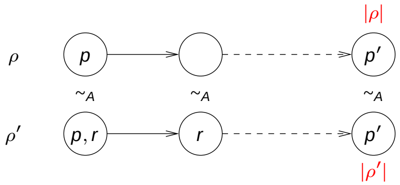
- State-based observability:

$$\rho_1 \sim_{Alice} \rho_2 \text{ if } \mathit{last\_state}(\rho_1) \sim_{Alice} \text{last\_}\mathit{state}(\rho_2)$$

- Good algorithmic properties (satisfiability, model-checking).
- Success tools: MCMAS (Lomuscio et. al), VerICS (Penczek).
- Forgetful semantics: agents don't remember anything their previous observations.

Semantics of temporal epistemic logics (3)

- Agent memory may play an essential role in knowledge acquisition.
- Indistinguishability for *Alice* on runs taking into consideration histories of observations:

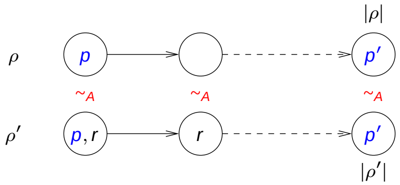


- Synchronous and perfect recall observability for *Alice*.
- Other variants: only synchronous, only perfect recall, and a dual of perfect recall called “no learning”.
- Concrete observability:
  - Some atomic propositions may be observed by some agents.

$$\Pi_A = \{p, p', \ldots\}$$

Semantics of temporal epistemic logics (3)

- Agent memory may play an essential role in knowledge acquisition.
- Indistinguishability for *Alice* on runs taking into consideration histories of observations:

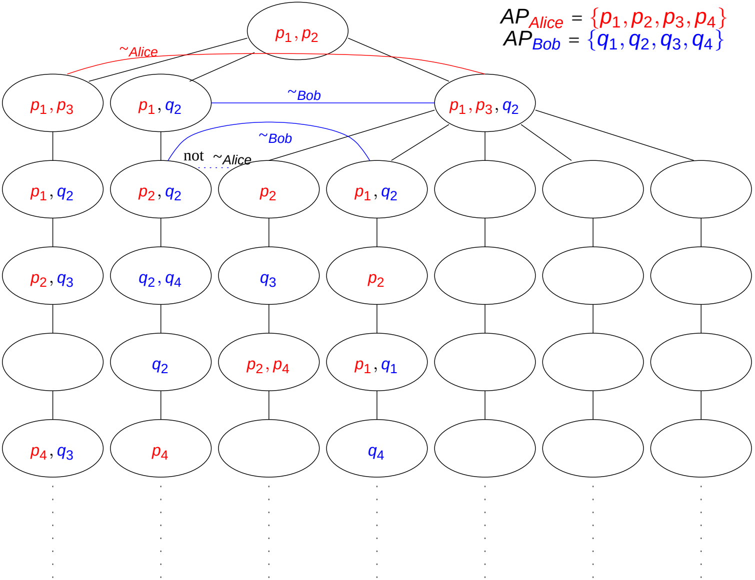


- Synchronous and perfect recall observability for *Alice*.
- Other variants: only synchronous, only perfect recall, and a dual of perfect recall called “no learning”.
- Concrete observability:
  - Some atomic propositions may be observed by some agents.

$$\Pi_A = \{p, p', \ldots\}$$

Synchronous & perfect recall semantics

$AP_{Alice} = \{p_1, p_2, p_3, p_4\}$
$AP_{Bob} = \{q_1, q_2, q_3, q_4\}$

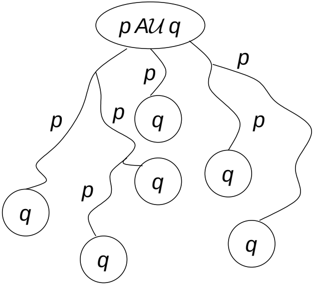


Semantics

$\mathcal{R}$: set of runs, $\rho \in \mathcal{R}$, i: position on the run.

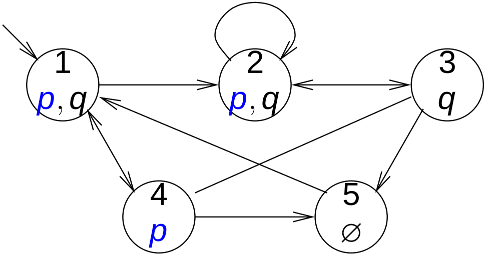
$$(\mathcal{R},\rho,i) \models K_A\phi \text{ if for any } \rho' \in \mathcal{R} \text{ with } \rho'[1..i] \sim_A \rho[1..i] \text{ we have that } (\mathcal{R},\rho',i) \models \phi$$

$$(\mathcal{R},\rho,i) \models p\,A\mathcal{U}\,q \text{ if for any } \rho' \in \mathcal{R} \text{ with } \rho'[1..i] = \rho[1..i] \text{ there exists } j \geq i \text{ with } (\mathcal{R},\rho',j) \models q \text{ and for all } i \leq k < j, (\mathcal{R},\rho',k) \models p$$



Synchronous & perfect recall vs. state-based

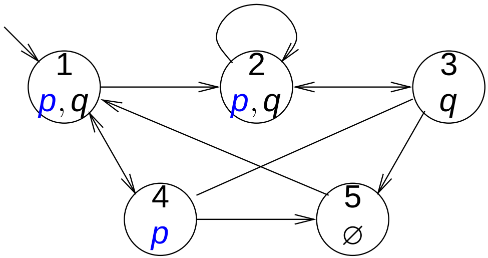
- Model-checking requires subset construction on the model:



- Suppose $\Pi_A = \{p\}$, hence $1 \sim_A 2 \sim_A 4$, $3 \sim_A 5$.
- Does $1 \to 2 \to 2 \models K_A q$? Yes:
  - $1 \to 2 \to 2 \models q$,
  - $1 \to 4 \to 1 \models q$, and no other identically-observable run!
- If we stick to state-based observability, then
  - $\ldots \to 2 \models q$, $\quad \ldots \to 1 \models q$, $\quad \ldots \to 4 \not\models q$,
  - ... so the answer is no!

Synchronous & perfect recall vs. state-based

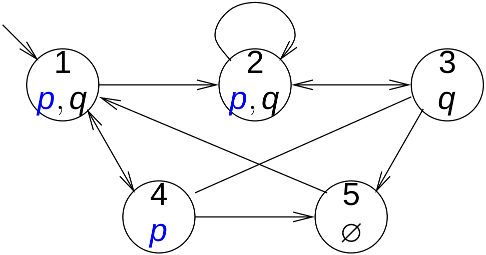
- Model-checking requires subset construction on the model:



- Suppose $\Pi_A = \{p\}$, hence $1 \sim_A 2 \sim_A 4$, $3 \sim_A 5$.
- Does $1 \to 2 \to 2 \models K_A q$? Yes:
  - $1 \to 2 \to 2 \models q$,
  - $1 \to 4 \to 1 \models q$, and no other identically-observable run!
- If we stick to state-based observability, then
  - $\ldots \to 2 \models q$, $\ldots \to 1 \models q$, $\ldots \to 4 \not\models q$,
  - ... so the answer is no!

Synchronous & perfect recall vs. state-based

- Model-checking requires subset construction on the model:



- Suppose $\Pi_A = \{p\}$, hence $1 \sim_A 2 \sim_A 4$, $3 \sim_A 5$.
- Does $1 \to 2 \to 2 \models K_A q$? Yes:
  - $1 \to 2 \to 2 \models q$,
  - $1 \to 4 \to 1 \models q$, and no other identically-observable run!
- If we stick to state-based observability, then
  - $\ldots \to 2 \models q$, $\quad \ldots \to 1 \models q$, $\quad \ldots \to 4 \not\models q$,
  - ... so the answer is no!

Model-checking epistemic LTL and CTL

	LTLK/CTLK	LTLK/CTLK with common knowledge
State-based obs.	PSPACE-complete	PSPACE-complete
Perf. recall & synch.	Nonelementary LTLK: v.d. Meyden & Shilov, '99 CTLK: Dima, '08	Undecidable

- Subset construction on the model for handling each knowledge operator.
- Nonelementary hardness still open.

Timed Computational Tree Logic

- CTL with clock formulas:

$$\phi ::= p \mid \mathcal{C} \mid \phi \wedge \phi \mid \neg\phi \mid \phi A\mathcal{U}\, \phi \mid \phi E\mathcal{U}\, \phi \mid z \text{ in } \phi$$
$$\mathcal{C} ::= z \in I \mid \mathcal{C} \wedge \mathcal{C} \mid \mathcal{C} \vee \mathcal{C}$$

  - $p \in \Pi$, set of atomic propositions.
  - $z \in \mathcal{X}$, set of clock variables, interpreted over $\mathbb{R}_{\geq 0}$.
  - $I \subseteq \mathbb{R}_{\geq 0}$ interval with integer (or infinite) bounds.

- Derived operators – timed until:

$$\phi A\mathcal{U}_I\, \psi = z \text{ in } \big(\phi A\mathcal{U}\, (z \in I \wedge \psi)\big)$$
$$\phi E\mathcal{U}_I\, \psi = z \text{ in } \big(\phi E\mathcal{U}\, (z \in I \wedge \psi)\big)$$

- Abbreviations:

$$E\Diamond_I\, \phi = \text{true}\, E\mathcal{U}_I\, \phi \qquad A\Diamond_I\, \phi = \text{true}\, A\mathcal{U}_I\, \phi$$
$$E\Box_I\, \phi = \neg A\Diamond_I\, \neg\phi \qquad \phi E\mathcal{U}\, \psi = \phi E\mathcal{U}_{[0,\infty[}$$

Timed Computational Tree Logic

- CTL with clock formulas:

$$\phi ::= p \mid \mathcal{C} \mid \phi \wedge \phi \mid \neg\phi \mid \phi\, A\mathcal{U}\, \phi \mid \phi\, E\mathcal{U}\, \phi \mid z \text{ in } \phi$$
$$\mathcal{C} ::= z \in I \mid \mathcal{C} \wedge \mathcal{C} \mid \mathcal{C} \vee \mathcal{C}$$

  - $p \in \Pi$, set of atomic propositions.
  - $z \in \mathcal{X}$, set of clock variables, interpreted over $\mathbb{R}_{\geq 0}$.
  - $I \subseteq \mathbb{R}_{\geq 0}$ interval with integer (or infinite) bounds.

- Derived operators – timed until:

$$\phi\, A\mathcal{U}_I\, \psi = z \text{ in } \big(\phi\, A\mathcal{U} \,(z \in I \wedge \psi)\big)$$
$$\phi\, E\mathcal{U}_I\, \psi = z \text{ in } \big(\phi\, E\mathcal{U} \,(z \in I \wedge \psi)\big)$$

- Abbreviations:

$$E\Diamond_I\, \phi = \text{true}\, E\mathcal{U}_I\, \phi \qquad A\Diamond_I\, \phi = \text{true}\, A\mathcal{U}_I\, \phi$$
$$E\Box_I\, \phi = \neg A\Diamond_I\, \neg\phi \qquad \phi\, E\mathcal{U}\, \psi = \phi\, E\mathcal{U}_{[0,\infty[}$$

Semantics of TCTL

- Continuous description of
  - Clock values.
  - Atomic proposition valuations.
- State trajectory T: at each time instant $t \in \mathbb{R}_{\geq 0}$,
  - $T_\Pi(t) : \Pi \to \{0,1\}$
  - $T_{clock}(t) : \mathcal{X} \to \mathbb{R}_{\geq 0}$.
- But clocks may be reset.
  - Weakly monotonic time.

Trajectories defined

State trajectory over Π

$T = (\mathcal{I}, \theta)$, where

1. $\mathcal{I} = \left(S_i, I_i\right)_{1 \leq i < \eta}$ time frame interpretation of atomic symbols.
   - $I_i = [\alpha_{i-1}, \alpha_i]$ closed interval.
   - S_i: atomic propositions interpreted as true along interval I_i.
2. $\theta = (\theta_i)_{1 \leq i < \eta}$
   - $\theta_i : [\alpha_{i-1}, \alpha_i] \times \mathcal{X} \to \mathbb{R}_{\geq 0}$.

$$\theta_i(t', x) = \theta(t, x) + t' - t,$$
$$\theta_{i+1}(\alpha_i, \cdot) = \theta_i(\alpha_i, \cdot)[X := 0] \text{ for some } X$$

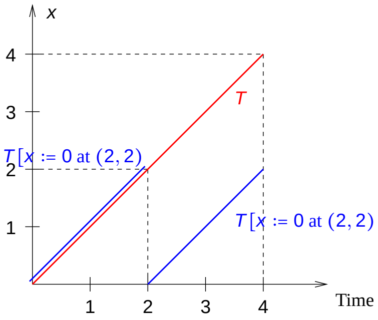
- Weakly monotonic time: $I_i \cap I_{i+1} = \alpha_i$.

Operations on trajectories

- Point on trajectory T: (k, β)
  - $1 \le k \le \eta$ = index of an interval.
  - $\beta \in [\alpha_{k-1}, \alpha_k]$, the actual time point.
- Total order on points: $(k, \beta) \prec (k', \beta')$ if
  - Either $k < k'$,
  - ... or $k = k'$ and $\beta < \beta'$.
- Resetting a clock at point (k, β): $T[x := 0 \text{ at } (k, \beta)]$.
- Prefix: $T[0..(k, \beta)]$.
- Concatenation.
- Projection: replace S_i with $S_i' = S_i \cap P$, for some $P \subseteq \Pi$.
- Length: $len(T) = \alpha_\eta$.

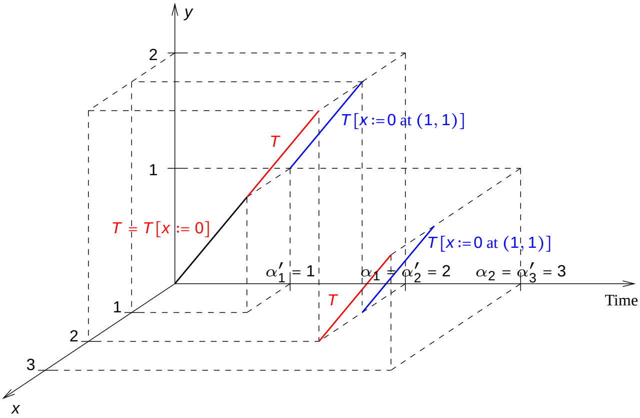
Trajectories (2)

- Clocks increase at rate 1 with time passage.
- At some points clocks may be reset.



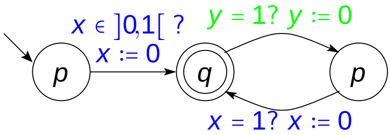
- $T[x := 0 \text{ at } (2,2)]$: trajectory resulting from T when clock x is reset at point $(2,2)$.

Trajectories (3)



Timed automata

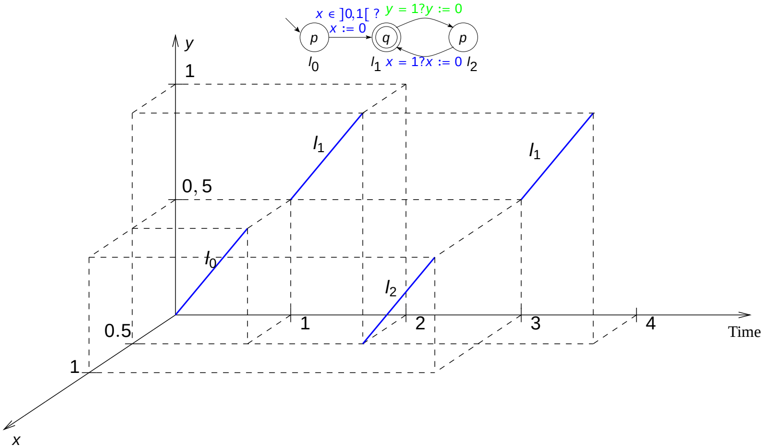
- Timed automata = automata with clocks for measuring time passage.
  - Clocks evolve synchronously.
  - Transitions guarded by simple clock constraints $x \in I$.



- Atomic propositions labeling automata locations.

Timed automata semantics

- Run = clock trajectory over the set of locations.



- Each trajectory over locations induces a trajectory over states.

TCTL semantics

R run in the automaton $\mathcal{A}$, (k,β) point on the *Time* axis of the run.

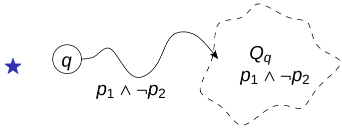
- $(R,k,\beta) \models C$ if $v \models C$ where v is the clock valuation at (k,β) in R.
- $(R,k,\beta) \models p$ if $p \in S_k$.
- $(R,k,\beta) \models z \text{ in } \phi$ if $(R[z:=0 \text{ at } (k,\beta)],k,\beta) \models \phi$.
- $(R,k,\beta) \models \phi_1 E\mathcal{U}\,\phi_2$ if
  - There exists some run $R' = (\mathcal{I}',\rho')$ for which $R[0..(k,\beta)] = R'[0..(k,\beta)]$
  - There exists a w-point (k',β') with $(k',\beta') \geq (k,\beta)$ and $(R',k',\beta') \models \phi_2$
  - And for all (k'',β'') for which $(k,\beta) \leq (k'',\beta'') < (k',\beta')$, we have that $(R',k'',\beta'') \models \phi_1$.

Exact translation of discrete semantics of $A\mathcal{U}$.

TCTL model-checking

State labeling algorithm on the region graph:

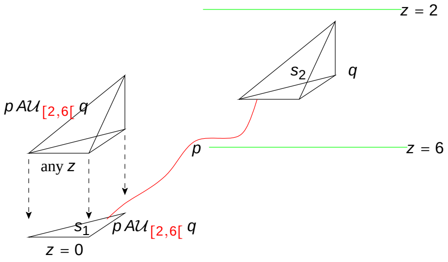
- Given formula ϕ, **split** the set of regions Reg into Reg_ϕ and $Reg_{\neg\phi}$
- Structural induction on the syntactic tree of ϕ.
- Add a new propositional symbol p_ϕ for each analyzed ϕ.
  - Label Q_ϕ with p_ϕ and do not label $Q_{\neg\phi}$ with p_ϕ.
- $\phi = p_1 A\mathcal{U} p_2$
  - $Reg_{\neg(p_1 A\mathcal{U} p_2)}$ contains q iff $\exists Reg_q \subseteq Reg$ strongly connected s.t.:



  - $Reg_{p_1 A\mathcal{U} p_2} = Reg \setminus Reg_{\neg(p_1 A\mathcal{U} p_2)}$.

Model-checking for freeze quantifiers

- Utilize a new clock z for checking the time bound in $p\,\mathsf{A}\mathcal{U}_I\,q$



- Essential trick: z can be reused for other subformulae.

TCTLK syntax

$$\phi ::= p \mid \mathcal{C} \mid \phi \wedge \phi \mid \neg\phi \mid \phi A\mathcal{U}\phi \mid \phi E\mathcal{U}\phi \mid z \text{ in } \phi \mid K_A\phi$$
$$\mathcal{C} ::= x \in I \mid \mathcal{C} \wedge \mathcal{C} \mid \mathcal{C} \vee \mathcal{C}$$

- $p \in \Pi$, set of atomic propositions,
- $x, z \in \mathcal{X}$, set of clock variables, interpreted over $\mathbb{R}_{\geq 0}$.
- $I \subseteq \mathbb{R}_{\geq 0}$ interval with integer (or infinite) bounds.

TCTLK semantics

- $\mathcal{A}$ timed automaton, with clocks $\mathcal{X}$
- For all $A \in Ag$
  1. $\Pi_A \subseteq \Pi$ – atoms whose truth value can be observed by A.
  2. $\mathcal{X}_A \subseteq \mathcal{X}$ – clocks whose value can be observed by A.
- Observability relation: runs R, R' in $\mathcal{A}$, $R \sim_A R'$ if:
  1. Both have the same length, $len(R) = len(r')$.
  2. $R\big|_{\Pi_A, \mathcal{X}_A} = R'\big|_{\Pi_A, \mathcal{X}_A}$.

TCTLK semantics

- $\mathcal{A}$ timed automaton, with clocks $\mathcal{X}$
- For all $A \in Ag$
  1. $\Pi_A \subseteq \Pi$ – atoms whose truth value can be observed by A.
  2. $\mathcal{X}_A \subseteq \mathcal{X}$ – clocks whose value can be observed by A.
- Observability relation: runs R, R' in $\mathcal{A}$, $R \sim_A R'$ if:
  1. Both have the same length, $len(R) = len(r')$.
  2. $R\big|_{\Pi_A, \mathcal{X}_A} = R'\big|_{\Pi_A, \mathcal{X}_A}$.

TCTLK semantics (2)

- Semantics of K_A:
  - $(R, k, \beta) \models K_A \phi$ if for any run R' and any w-point (k', β) in R' with $\text{traj}(R')[0..(k', \beta)]\big|_{\Pi_A, \mathcal{X}_A} = \text{traj}(R)[0..(k, \beta)]\big|_{\Pi_A, \mathcal{X}_A}$ we have that $(R', k', \beta) \models \phi$.
- Needs a complete description of the trajectory (clocks & states).
  - And weakly monotonic time!

TCTLK semantics (2)

- Semantics of K_A:
  - $(R, k, \beta) \models K_A\phi$ if for any run R' and any w-point (k', β) in R' with $\mathrm{traj}(R')[0..(k', \beta)]\big|_{\Pi_A, \mathcal{X}_A} = \mathrm{traj}(R)[0..(k, \beta)]\big|_{\Pi_A, \mathcal{X}_A}$ we have that $(R', k', \beta) \models \phi$.
- Needs a complete description of the trajectory (clocks & states).
  - And weakly monotonic time!

Some example formula

$$\big(P_A\, E\Diamond_{\leq 3}\, danger\big) \wedge \neg E\Diamond_{\leq 3}\, danger$$

- Observations for sensor *A* would indicate that it's possible that in less than 3 time units the system goes into a dangerous state
- But this is not the case in the current state.

Model-checking, general case

Theorem

Model-checking is undecidable for TCTLK with timed automata semantics.

- Subset construction needed for the knowledge modalities.
- ... but timed automata are not determinizable!

An even stronger result:

Theorem

Model-checking is undecidable for CTLK with timed automata semantics.

That is, no clock formulas, no freeze quantifiers!

Model-checking, general case

Theorem

Model-checking is undecidable for TCTLK with timed automata semantics.

- Subset construction needed for the knowledge modalities.
- ... but timed automata are not determinizable!

An even stronger result:

Theorem

Model-checking is undecidable for CTLK with timed automata semantics.

That is, no clock formulas, no freeze quantifiers!

Undecidability of the model-checking problem

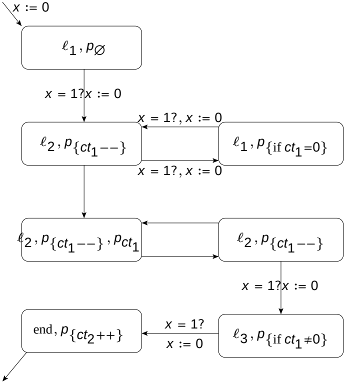
- Reduction to the halting problem for 2-counter (Minsky) machines.
- Configuration (ℓ, ct_1, ct_2) coded as a unit-length part of a trajectory, during which
  - ct_1 points where p_{ct_1} holds, and
  - ct_2 points where p_{ct_2} holds.
- Coding done partly in the timed automaton, partly in the formula.
  - The timed automaton simulates, along some trajectories, the control flow in the 2-counter program,
  - Each encoding of a configuration (ℓ, ct_1, ct_2) can be followed, in the timed automaton, by an encoding of the next configuration (ℓ', ct_1', ct_2').
  - The TCTLK formula is used to show that each run R which encodes the run $\theta[1..n]$ of the 2-counter machine, can be extended to another run R' which encodes $\theta[1..n+1]$.

Undecidability of the model-checking problem (2)

A simple 2-counter program:

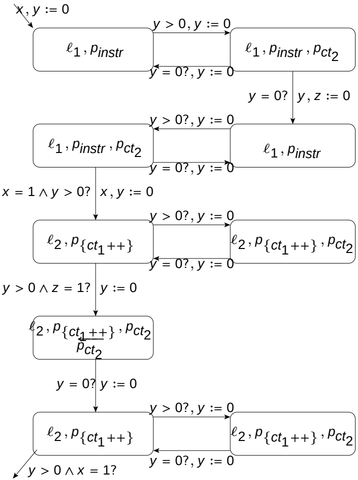
$\ell_1 \quad : ct_1 --$
$\ell_2 \quad : \text{if } ct_1 = 0 \text{ goto } \ell_1$
$\ell_3 \quad : ct_2 ++$

Modeling the control flow:



Undecidability of the model-checking problem (3)

Copying some counter value:



Undecidability of the model-checking problem (4)

Extending a run:

$$E\Box\Big(\big(\ell_2 \wedge p_{\{ct_1++\}} \wedge p_{ct_2} \rightarrow P_A(\overleftarrow{p_{ct_2}})\big)\wedge$$

$$\big(\ell_2 \wedge p_{\{ct_1++\}} \wedge \neg p_{ct_2} \rightarrow P_A(\overleftarrow{p_{\neg ct_2}})\big)\Big)\wedge$$

$$E\Box\Big(\big(\ell_2 \wedge p_{\{ct_1++\}} \wedge p_{ct_1} \wedge \neg p_{ct_1}^{last} \rightarrow P_A(\overleftarrow{p_{ct_1}})\big)\wedge$$

$$\big(\ell_2 \wedge p_{\{ct_1++\}} \wedge (p_{ct_1}^{last} \vee \neg p_{ct_1}) \rightarrow P_A(\overleftarrow{p_{\neg ct_1}})\big)\Big)\wedge$$

$$E\Box\Big(\big(\ell_2 \wedge p_{\{ct_1--\}} \wedge (p_{ct_1}^{last} \vee p_{ct_1}) \rightarrow P_A(\overleftarrow{p_{ct_1}^{last}})\big)\wedge$$

$$\big(\ell_2 \wedge p_{\{ct_1--\}} \wedge p_{\neg ct_1} \rightarrow P_A(\overleftarrow{p_{\neg ct_1}})\big)\Big)$$

Model-checking, special case

- Full observability of clock values: $\mathcal{X}_A = \mathcal{X}$ for all agents $A \in Ag$.

Theorem

The model-checking problem for TCTLK with full observability of clock values is decidable.

- With full observability of clock values, the subset construction needed for K_A only needs to be done on locations!

Conclusions

- A concrete semantics of observability.
  - Each agent can observe truth values of some atomic propositions.
  - ... and some clock values.
  - Synchronous & perfect recall semantics.
- Partial clock observability $\mapsto$ undecidable model-checking.
  - Subset construction needed for treating K_A.
  - Undecidability even in the absence of clock formulas and freeze quantifiers.
- Full clock observability $\mapsto$ decidable model-checking.

Further work

Short term:

- Restricting the semantics to determinizable subclasses of timed automata.
- Implementation, case studies, etc.

Long term:

- *Approximate* observability in presence of clock drift.
- Semantics of real-time knowledge-based programming languages.
- Implementation of real-time knowledge-based specifications.
- "Pure" TCTL model-checking with stopwatch automata.
- Real-time distributed controller synthesis with partial observation.
- Knowledge (individual/group) as assume-guarantee reasoning.