Model Checking: the Interval Way

Angelo Montanari
Dept. of Mathematics, Computer Science, and Physics
University of Udine, Italy
(On leave at UWA - January-February 2018)

Guest Speaker Research Seminar
School of Physics, Mathematics and Computing
The University of Western Australia in Perth
February 21, 2018

Model checking

Model checking: the desired properties of a system are checked against a model of it

- the model is usually a (finite) state-transition system
- system properties are specified by a temporal logic (LTL, CTL, CTL* and the like)

Distinctive features of model checking:

- exaustive check of all the possible behaviours
- fully automatic process
- a counterexample is produced for a violated property

Point-based vs. interval-based model checking

Model checking is usually point-based:

- properties express requirements over points (snapshots) of a computation (states of the state-transition system)
- they are specified by means of point-based temporal logics such as LTL, CTL, and CTL*

Interval properties express conditions on computation stretches instead of on computation states

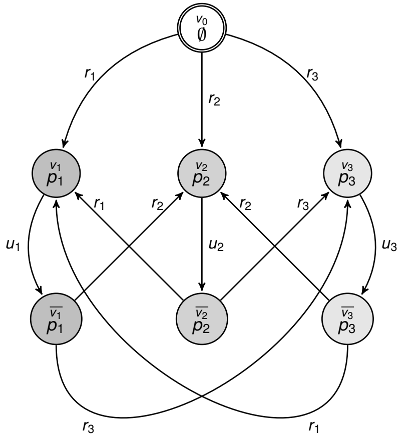
A lot of work has been done on interval temporal logic (ITL) satisfiability checking (an up-to-date survey can be found at: $https://users.dimi.uniud.it/\sim angelo.montanari/Movep$2016-$partI.pdf$).

ITL model checking entered the research agenda only recently (Bozzelli, Lomuscio, Michaliszyn, Molinari, Montanari, Murano, Perelli, Peron, Sala)

Outline of the talk

- The model checking problem for interval temporal logics
- Complexity results: the general picture
- Interval vs. point temporal logic model checking: an expressiveness comparison (a short account)
- Interval temporal logic model checking with regular expressions (a short account)
- Ongoing work and future developments

The modeling of the system: Kripke structures



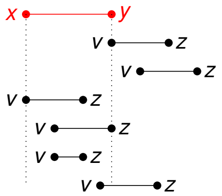
An example of Kripke structure

- HS formulas are interpreted over (finite) state-transition systems, whose states are labeled with sets of proposition letters (Kripke structures)
- An interval is a trace (finite path) in a Kripke structure

HS: the modal logic of Allen's interval relations

Allen's interval relations: the 13 binary ordering relations between 2 intervals on a linear order. They give rise to corresponding unary modalities over frames where intervals are primitive entities:

- HS features a modality for any Allen ordering relation between pairs of intervals (except for equality)

Allen rel.	HS	Definition	Example
meets	$\langle A \rangle$	$[x,y]\mathcal{R}_A[v,z] \iff y = v$	
before	$\langle L \rangle$	$[x,y]\mathcal{R}_L[v,z] \iff y < v$	
started-by	$\langle B \rangle$	$[x,y]\mathcal{R}_B[v,z] \iff x = v \wedge z < y$	
finished-by	$\langle E \rangle$	$[x,y]\mathcal{R}_E[v,z] \iff y = z \wedge x < v$	
contains	$\langle D \rangle$	$[x,y]\mathcal{R}_D[v,z] \iff x < v \wedge z < y$	
overlaps	$\langle O \rangle$	$[x,y]\mathcal{R}_O[v,z] \iff x < v < y < z$	

All modalities can be expressed by means of $\langle A \rangle$, $\langle B \rangle$, $\langle E \rangle$, and their transposed modalities only (if point intervals are admitted, $\langle B \rangle$, $\langle E \rangle$, and their transposed modalities suffice)

HS semantics and model checking

Truth of a formula ψ over a trace ρ of a Kripke structure $\mathcal{K} = (\mathcal{AP}, W, \delta, \mu, w_0)$ defined by induction on the complexity of ψ:

- $\mathcal{K}, \rho \models p$ iff $p \in \bigcap_{w \in \mathrm{states}(\rho)} \mu(w)$, for any letter $p \in \mathcal{AP}$ (homogeneity assumption);
- clauses for negation, disjunction, and conjunction are standard;
- $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi$ iff there is a trace ρ' s.t. $\mathrm{lst}(\rho) = \mathrm{fst}(\rho')$ and $\mathcal{K}, \rho' \models \psi$;
- $\mathcal{K}, \rho \models \langle \mathrm{B} \rangle \psi$ iff there is a proper prefix ρ' of ρ s.t. $\mathcal{K}, \rho' \models \psi$;
- $\mathcal{K}, \rho \models \langle \mathrm{E} \rangle \psi$ iff there is a proper suffix ρ' of ρ s.t. $\mathcal{K}, \rho' \models \psi$;
- the semantic clauses for $\langle \overline{\mathrm{A}} \rangle$, $\langle \overline{\mathrm{B}} \rangle$, and $\langle \overline{\mathrm{E}} \rangle$ are similar

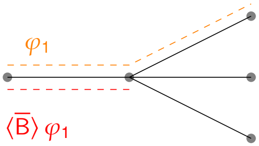
Model Checking

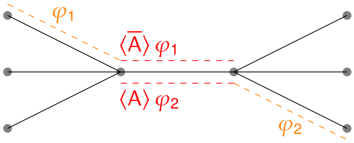
$\mathcal{K} \models \psi \iff$ for all initial traces ρ of $\mathcal{K}$, it holds that $\mathcal{K}, \rho \models \psi$

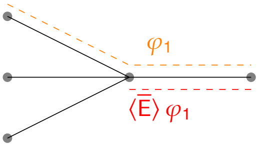
Possibly infinitely many traces!

Remark: HS state semantics (HS_{st})

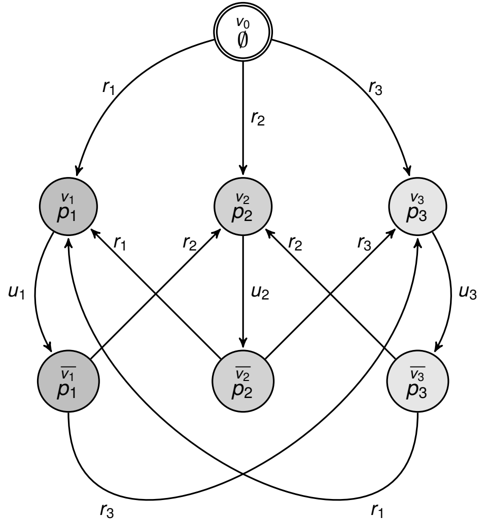
- According to the given semantics, HS modalities allow one to branch both in the past and in the future




φ_1
$\langle\overline{A}\rangle\,\varphi_1$
$\langle A\rangle\,\varphi_2$
φ_2



The Kripke structure $\mathcal{K}_{Sched}$ for a simple scheduler



A short account of $\mathcal{K}_{Sched}$

$\mathcal{K}_{Sched}$ models the behaviour of a scheduler serving 3 processes which are continuously requesting the use of a common resource (it can be easily generalised to an arbitrary number of processes)

Initial state: v_0 (no process is served in that state)

In v_i and $\overline{v}_i$ the i-th process is served (p_i holds in those states)

The scheduler cannot serve the same process twice in two successive rounds:

- process i is served in state v_i, then, after “some time”, a transition u_i from v_i to $\overline{v}_i$ is taken; subsequently, process i cannot be served again immediately, as v_i is not directly reachable from $\overline{v}_i$
- a transition r_j, with $j \neq i$, from $\overline{v}_i$ to v_j is then taken and process j is served

Some meaningful properties to be checked over $\mathcal{K}_{Sched}$

Validity of properties over all legal computation intervals can be forced by modality $[E]$ (they are suffixes of at least one initial trace)

Property 1: in any computation interval of length at least 4, at least 2 processes are witnessed (YES/no process can be executed twice in a row)

$$\mathcal{K}_{Sched} \models [E]\big(\langle \mathrm{E}\rangle^3 \top \rightarrow (\chi(p_1,p_2) \vee \chi(p_1,p_3) \vee \chi(p_2,p_3))\big),$$

where $\chi(p,q) = \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle q$
Property 2: in any computation interval of length at least 11, process 3 is executed at least once (NO/the scheduler can postpone the execution of a process ad libitum—starvation)

$$\mathcal{K}_{Sched} \not\models [E](\langle \mathrm{E}\rangle^{10} \top \rightarrow \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_3)$$

Property 3: in any computation interval of length at least 6, all processes are witnessed (NO/the scheduler should be forced to execute them in a strictly periodic manner, which is not the case)

$$\mathcal{K}_{Sched} \not\models [E](\langle \mathrm{E}\rangle^{5} \rightarrow (\langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_1 \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_2 \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_3))$$

Model checking: the key notion of BE_k-descriptor

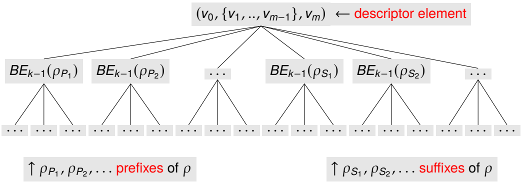
- The BE-nesting depth of an HS formula ψ ($\mathrm{Nest}_{\mathrm{BE}}(\psi)$) is the maximum degree of nesting of modalities B and E in ψ
- Two traces ρ and ρ' of a Kripke structure $\mathcal{K}$ are k-equivalent if and only if $\mathcal{K}, \rho \models \psi$ iff $\mathcal{K}, \rho' \models \psi$ for all HS-formulas ψ with $\mathrm{Nest}_{\mathrm{BE}}(\psi) \leq k$

Model checking: the key notion of BE_k-descriptor

- The BE-nesting depth of an HS formula ψ ($\mathrm{Nest}_{\mathrm{BE}}(\psi)$) is the maximum degree of nesting of modalities B and E in ψ
- Two traces ρ and ρ' of a Kripke structure $\mathcal{K}$ are k-equivalent if and only if $\mathcal{K}, \rho \models \psi$ iff $\mathcal{K}, \rho' \models \psi$ for all HS-formulas ψ with $\mathrm{Nest}_{\mathrm{BE}}(\psi) \leq k$

For any given k, we provide a suitable tree representation for a trace, called a BE_k-descriptor

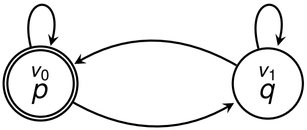
The BE_k-descriptor for a trace $\rho = v_0 v_1 .. v_{m-1}\ v_m$, denoted $BE_k(\rho)$, has the following structure:



$(v_0, \{v_1, .., v_{m-1}\}, v_m)$ ← descriptor element
$BE_{k-1}(\rho_{P_1})$
$BE_{k-1}(\rho_{P_2})$
...
$BE_{k-1}(\rho_{S_1})$
$BE_{k-1}(\rho_{S_2})$
...
↑ $\rho_{P_1}, \rho_{P_2}, \ldots$ prefixes of ρ
↑ $\rho_{S_1}, \rho_{S_2}, \ldots$ suffixes of ρ

Remark: the descriptor does not feature sibling isomorphic subtrees

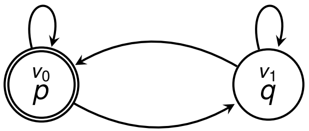
An example of a BE_2-descriptor



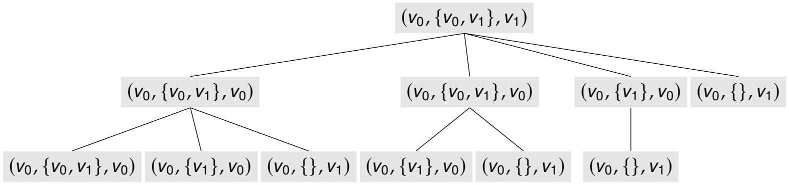
The BE_2-descriptor for the trace $\rho = v_0 v_1 v_0^4 v_1$ (for the sake of readability, only the subtrees for prefixes are displayed and point intervals are excluded)

$$\begin{tikzcd}
 & & & (v_0,\{v_0,v_1\},v_1) \arrow[dll, dash] \arrow[d, dash] \arrow[drr, dash] \arrow[drrr, dash] & & & \\
 & (v_0,\{v_0,v_1\},v_0) \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & (v_0,\{v_0,v_1\},v_0) \arrow[d, dash] \arrow[dr, dash] & & (v_0,\{v_1\},v_0) \arrow[d, dash] & (v_0,\{\},v_1) \\
(v_0,\{v_0,v_1\},v_0) & (v_0,\{v_1\},v_0) & (v_0,\{\},v_1) & (v_0,\{v_1\},v_0) & (v_0,\{\},v_1) & (v_0,\{\},v_1) &
\end{tikzcd}$$

An example of a BE_2-descriptor



The BE_2-descriptor for the trace $\rho = v_0 v_1 v_0^4 v_1$ (for the sake of readability, only the subtrees for prefixes are displayed and point intervals are excluded)



Remark: the subtree to the left is associated with both prefixes $v_0 v_1 v_0^3$ and $v_0 v_1 v_0^4$ (no sibling isomorphic subtrees in the descriptor)

Decidability of model checking for full HS

FACT 1: For any Kripke structure $\mathcal{K}$ and any BE-nesting depth $k \geq 0$, the number of different BE_k-descriptors is finite (and thus at least one descriptor has to be associated with infinitely many traces)

Decidability of model checking for full HS

FACT 1: For any Kripke structure $\mathcal{K}$ and any BE-nesting depth $k \geq 0$, the number of different BE_k-descriptors is finite (and thus at least one descriptor has to be associated with infinitely many traces)

FACT 2: Two traces ρ and ρ' of a Kripke structure $\mathcal{K}$ described by the same BE_k descriptor are k-equivalent

Decidability of model checking for full HS

FACT 1: For any Kripke structure $\mathcal{K}$ and any BE-nesting depth $k \geq 0$, the number of different BE_k-descriptors is finite (and thus at least one descriptor has to be associated with infinitely many traces)

FACT 2: Two traces ρ and ρ' of a Kripke structure $\mathcal{K}$ described by the same BE_k descriptor are k-equivalent

Theorem

The model checking problem for full HS *on finite Kripke structures is decidable (with a non-elementary algorithm)*

A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron, Checking Interval Properties of Computations, Acta Informatica, Special Issue: Temporal Representation and Reasoning (TIME'14), Vol. 56, n. 6-8, October 2016, pp. 587-619

Decidability of model checking for full HS

FACT 1: For any Kripke structure $\mathcal{K}$ and any BE-nesting depth $k \geq 0$, the number of different BE_k-descriptors is finite (and thus at least one descriptor has to be associated with infinitely many traces)

FACT 2: Two traces ρ and ρ' of a Kripke structure $\mathcal{K}$ described by the same BE_k descriptor are k-equivalent

Theorem

The model checking problem for full HS *on finite Kripke structures is decidable (with a non-elementary algorithm)*

A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron, Checking Interval Properties of Computations, Acta Informatica, Special Issue: Temporal Representation and Reasoning (TIME'14), Vol. 56, n. 6-8, October 2016, pp. 587-619

What about lower bounds?

The logic BE

Theorem

The model checking problem for BE, *over finite Kripke structures, is EXPSPACE-hard*

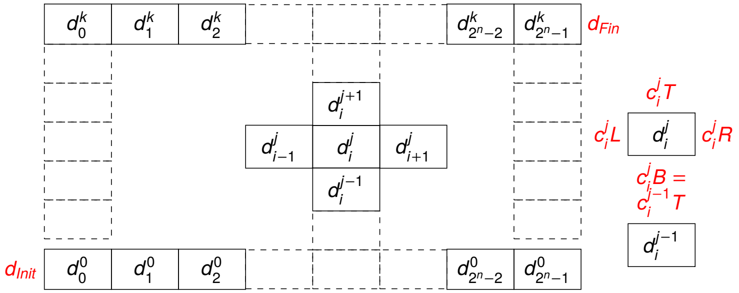
L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala, Interval Temporal Logic Model Checking: The Border Between Good and Bad HS Fragments, IJCAR 2016

Proof: a polynomial-time reduction from a domino-tiling problem for grids with rows of single exponential length

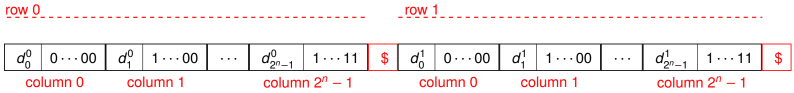
- for an instance $\mathcal{I}$ of the problem, we build a Kripke structure $\mathcal{K}_{\mathcal{I}}$ and a BE formula $\varphi_{\mathcal{I}}$ in polynomial time
- there is an initial trace of $\mathcal{K}_{\mathcal{I}}$ satisfying $\varphi_{\mathcal{I}}$ iff there is a tiling of $\mathcal{I}$
- $\mathcal{K}_{\mathcal{I}} \models \neg\varphi_{\mathcal{I}}$ iff there exists no tiling of $\mathcal{I}$

BE hardness: encoding of the domino-tiling problem

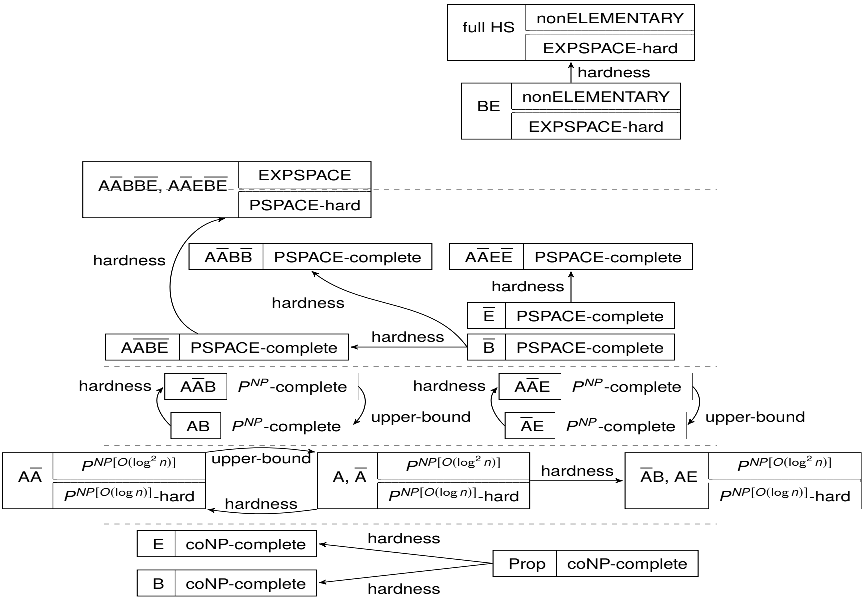
Instance of the tiling problem: $(C, \Delta, n, d_{init}, d_{final})$, with C a finite set of colors and $\Delta \subseteq C \times C \times C \times C$ a set of tuples (c_B, c_L, c_T, c_R)



String (interval) encoding of the problem



The complexity picture



Three main gaps to fill

There are three main gaps to fill:

- full HS and BE are in between nonELEMENTARY and EXPSPACE
- $A\overline{A}B\overline{BE}$, $A\overline{A}E\overline{BE}$, $AB\overline{BE}$, $AE\overline{BE}$, $\overline{A}B\overline{BE}$, and $\overline{A}E\overline{BE}$ are in between EXPSPACE and PSPACE
- A, $\overline{A}$, $A\overline{A}$, $\overline{A}B$, and AE are in between $P^{NP[O(\log^2 n)]}$ and $P^{NP[O(\log n)]}$

The first gap is definitely the most significant one (a progress report)

Point vs. interval temporal logic model checking

Question: is there any advantage in replacing points by intervals as the primary temporal entities, or is it just a matter of taste?

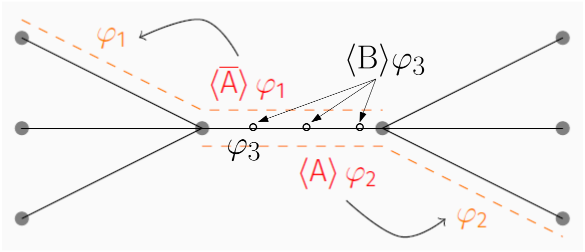
In order to compare the expressiveness of HS in model checking with those of LTL, CTL, and CTL*, we consider three semantic variants of HS:

- HS with state-based semantics (the original one)
- HS with computation-tree-based semantics
- HS with trace-based semantics

These variants are compared with the above-mentioned standard temporal logics and among themselves

L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala, Interval vs. Point Temporal Logic Model Checking: an Expressiveness Comparison. Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), December 2016, pp. 26:1-14

Branching semantic variant of HS



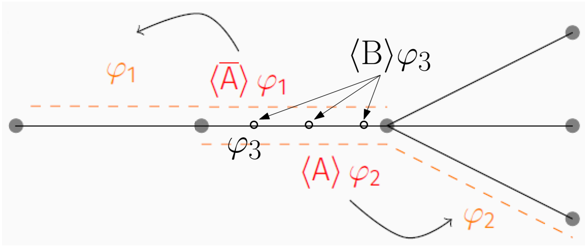
φ_1
$\langle \overline{A} \rangle \varphi_1$
$\langle B \rangle \varphi_3$
φ_3
$\langle A \rangle \varphi_2$
φ_2

State-based semantics of HS (HS_{st}):

- both the future and the past are branching

A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron, Checking Interval Properties of Computations, Acta Informatica, Special Issue: Temporal Representation and Reasoning (TIME'14), Vol. 56, n. 6-8, October 2016, pp. 587-619

Linear-past semantic variant of HS



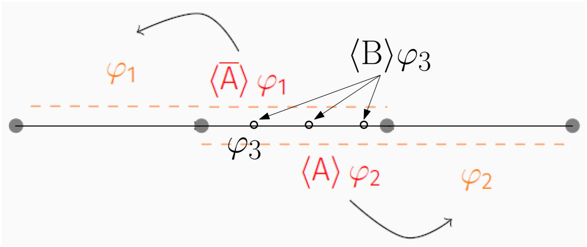
φ_1
$\langle \overline{A} \rangle\ \varphi_1$
$\langle B \rangle \varphi_3$
φ_3
$\langle A \rangle\ \varphi_2$
φ_2

Computation-tree-based semantics of HS (HS_{ct}):

- the future is branching
- the past is linear, finite and cumulative
- similar to CTL* + linear past

A. Lomuscio and J. Michaliszyn, Decidability of model checking multi-agent systems against a class of EHS specifications, Proc. of the 21st European Conference on Artificial Intelligence (ECAI), August 2014, pp. 543–548

Linear semantic variant of HS


φ_1
$\langle \overline{A} \rangle \varphi_1$
$\langle B \rangle \varphi_3$
φ_3
$\langle A \rangle \varphi_2$
φ_2

Trace-based semantics of HS (HS_{lin}):

- neither the past not the future is branching
- similar to LTL + past

The expressiveness picture

$$\begin{tikzcd}
\boxed{HS_{lin}} \arrow[r, dash, dashed, "\equiv"] \arrow[d, dash, dashed, "\neq"] \arrow[dd, dash, dashed, bend right, "\neq"'] & \boxed{LTL} & \\
\boxed{HS_{lp}} \arrow[r, dash, dashed, "\equiv"] \arrow[d, dash, dashed, "\wedge"] \arrow[dr, dash, dashed, "\neq"] & \boxed{\text{finitary } CTL*} \arrow[r, dash, dashed, "<"] & \boxed{CTL*} \\
\boxed{HS_{st}} \arrow[r, dash, dashed, "\neq"'] \arrow[urr, dash, dashed, bend right, "\neq"'] & \boxed{CTL} &
\end{tikzcd}$$

ITL model checking with regular expressions

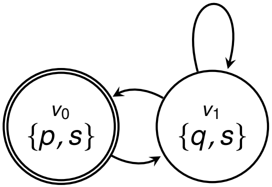
Can we relaxe the homogeneity assumption? The addition of regular expressions:

$$r ::= \varepsilon \mid \phi \mid r \cup r \mid r \cdot r \mid r^*$$

where ϕ is a Boolean (propositional) formula over $\mathcal{AP}$.

Examples:

- $r_1 = (\mathbf{p} \wedge \mathbf{s}) \cdot \mathbf{s}^* \cdot (\mathbf{p} \wedge \mathbf{s})$
- $r_2 = (\neg \mathbf{p})^*$


$$\begin{tikzcd} \boxed{\substack{v_0 \\ \{p,s\}}} \arrow[r, bend right] & \boxed{\substack{v_1 \\ \{q,s\}}} \arrow[l, bend right] \arrow[loop above] \end{tikzcd}$$

- $\rho = v_0 v_1 v_0 v_1 v_1$
- $\mu(\rho) = \{p,s\}\{q,s\}\{p,s\}\{q,s\}\{q,s\}$
- $\rho' = v_0 v_1 v_1 v_1 v_0$
- $\mu(\rho') = \{p,s\}\{q,s\}\{q,s\}\{q,s\}\{p,s\}$
  - $\mu(\rho) \notin \mathcal{L}(r_1)$, but $\mu(\rho') \in \mathcal{L}(r_1)$
  - $\mu(\rho) \notin \mathcal{L}(r_2)$ and $\mu(\rho') \notin \mathcal{L}(r_2)$

ITL model checking with regular expressions

In the definition of the truth of a formula ψ over a trace ρ of a Kripke structure $\mathcal{K} = (\mathcal{AP}, W, \delta, \mu, w_0)$, we replace the clause for propositional letters by a clause for regular expressions:

- $\mathcal{K}, \rho \models r$ iff $\mu(\rho) \in \mathcal{L}(r)$

Homogeneity can be recovered as a special case. To force it, all regular expressions in the formula must be of the form:

$$p \cdot (p)^*$$

Solution: given $\mathcal{K}$ and an HS formula φ over $\mathcal{AP}$, we build an NFA over $\mathcal{K}$ accepting the set of traces ρ such that $\mathcal{K}, \rho \models \varphi$.

ITL model checking with regular expressions

Model checking for full HS with regular expressions is decidable and its complexity, when restricted to system models—that is, if we assume the formula to be constant length—is PTIME

Model checking for $A\overline{A}B\overline{B}$ and its fragments is PSPACE-complete

L. Bozzelli, A. Molinari, A. Montanari, and A. Peron, An In-Depth Investigation of Interval Temporal Logic Model Checking with Regular Expressions. Proc. of the 15th International Conference on Software Engineering and Formal Methods (SEFM), LNCS 10469, Springer, September 2017, pp. 104-119

Model checking for $A\overline{A}B\overline{B}\overline{E}$ and $A\overline{A}E\overline{B}\overline{E}$ with regular expressions is $\mathbf{AEXP_{pol}}$-complete ($\mathbf{AEXP_{pol}}$ is the complexity class of problems decided by exponential-time bounded alternating Turing Machines with a polynomially bounded number of alternations)

L. Bozzelli, A. Molinari, A. Montanari, and A. Peron, On the Complexity of Model Checking for Syntactically Maximal Fragments of the Interval Temporal Logic HS with Regular Expressions. Proc. of the 8th International Symposium on Games, Automata, Logics and Formal Verification (GandALF), EPTCS 256, September 2017, pp. 31-45

Ongoing work and future developments - 1

Ongoing work: to determine the exact complexity of the satisfiability / model checking problems for BE over finite linear orders, under the homogeneity assumption (the three semantic variants of HS coincide over BE)

We know that the satisfiability/model checking problems for D over finite linear orders, under the homogeneity assumption, are PSPACE-complete (we exploit a spatial encoding of the models for D and a suitable contraction technique)

L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala, Satisfiability and Model Checking for the Logic of Sub-Intervals under the Homogeneity Assumption, Proc. of the 44th International Colloquium on Automata, Languages, and Programming(ICALP), LIPIcs 80, July 2017, pp. 120:1–120:14

There is no a natural way to generalize the solution for D to BE

Ongoing work and future developments - 2

Ongoing work: we are looking for possible replacements of Kripke structures by more expressive system models

- visibly pushdown systems, that can encode recursive programs and infinite state systems
- inherently interval-based models, that allows one to directly describe systems on the basis of their interval behavior/properties, such as, for instance, those involving actions with duration, accomplishments, or temporal aggregations (no restriction on the evaluation of proposition letters)

As for future developments: planning as satisfiability checking / model checking in interval temporal logic