Deterministic temporal logics and interval constraints (part 2)

Kamal Lodaya and Paritosh Pandya
Work with Simoni Shah, A.V. Sreejith

The Institute of Mathematical Sciences, Chennai, and
Tata Institute of Fundamental Research, Mumbai

January 2017

Linear temporal logic *LTL*

$\alpha ::= p \in Prop \mid \neg\, \alpha \mid \alpha \ \vee \ \beta \mid \ \bigcirc \alpha \mid \Diamond \alpha \mid \phi \ \mathsf{U} \ \beta$

Given a state sequence $\sigma : \mathbb{N} \to \wp(Prop)$ or word over the alphabet $A = \wp(Prop)$. (This is called a model.)

- $\sigma, i \models p$ iff $p \in \sigma(i)$
- $\sigma, i \models \bigcirc \alpha$ iff $\sigma, i+1 \models \alpha$
- $\sigma, i \models \alpha \,\mathsf{U}\, \beta$ iff for some $k : i \leq k : \sigma, k \models \beta$ and for all $j : i \leq j < k : \sigma, j \models \alpha$

Linear temporal logic *LTL*

$\alpha ::= p \in Prop \mid \neg\, \alpha \mid \alpha \ \vee \ \beta \mid \ \bigcirc \alpha \mid \Diamond \alpha \mid \phi \ \mathsf{U} \ \beta$

Given a state sequence $\sigma : \mathbb{N} \to \wp(Prop)$ or word over the alphabet $A = \wp(Prop)$. (This is called a model.)

- $\sigma, i \models p$ iff $p \in \sigma(i)$
- $\sigma, i \models \bigcirc \alpha$ iff $\sigma, i+1 \models \alpha$
- $\sigma, i \models \alpha \,\mathsf{U}\, \beta$ iff for some $k : i \leq k : \sigma, k \models \beta$ and for all $j : i \leq j < k : \sigma, j \models \alpha$

The right hand side (semantics) is a formula of first order logic over $\mathbb{N}$. It uses three variables i, j, k.

$\Diamond \alpha$ is $true \,\mathsf{U}\, \alpha$. Its semantics is a formula of first order logic with two variables FO^2. Can also consider "past" modalities ($\diamondminus \alpha, \boxminus \alpha, \ominus \alpha$ mirrors of $\Diamond \alpha, \Box \alpha, \bigcirc \alpha$).

Modulo/threshold counting and group computation guards

Let $B, B_i \subseteq A$, let $c_i \in \mathbb{Z}$, let $t, u \in \mathbb{N}$, let $q \geq 2$.
Let $(G, +, 0) = \{h_1, \ldots, h_k, 0\}$ be a finite group.

$$
\begin{array}{lcl}
(simple)sg & ::= & \#B = 0 \\
modg & ::= & \Sigma_i c_i \#B_i \in R \bmod q, \text{ where } R \subseteq \{0, \ldots, q-1\}. \\
grpg & ::= & \Sigma_G(c_1\#B_1, \ldots, c_k\#B_k) \in H, \text{ where } H \subseteq G \\
thrg & ::= & t \sim \#B \mid \#B \sim u \mid t \sim \#B \sim' u, \\
 & & \qquad \text{where } \sim, \sim' \text{ in } \{<, \leq\} \\
ling & ::= & modg \mid thrg
\end{array}
$$

Abbreviations: $\Sigma_i c_i \#B_i \equiv r \bmod q$, $\#B = t$.

Modulo/threshold counting and group computation guards

Let $B, B_i \subseteq A$, let $c_i \in \mathbb{Z}$, let $t, u \in \mathbb{N}$, let $q \geq 2$.
Let $(G, +, 0) = \{h_1, \ldots, h_k, 0\}$ be a finite group.

$$
\begin{array}{lll}
(simple)sg & ::= & \#B = 0 \\
modg & ::= & \Sigma_i c_i \#B_i \in R \bmod q, \text{ where } R \subseteq \{0, \ldots, q-1\}. \\
grpg & ::= & \Sigma_G(c_1\#B_1, \ldots, c_k\#B_k) \in H, \text{ where } H \subseteq G \\
thrg & ::= & t \sim \#B \mid \#B \sim u \mid t \sim \#B \sim' u, \\
 & & \qquad \text{where } \sim, \sim' \text{ in } \{<, \leq\} \\
ling & ::= & modg \mid thrg
\end{array}
$$

Abbreviations: $\Sigma_i c_i \#B_i \equiv r \bmod q$, $\#B = t$.

Boolean combinations:

$$
\begin{array}{lll}
bsg & ::= & sg \mid \neg bsg \mid bsg_1 \vee bsg_2 \\
bmg & ::= & modg \mid \neg bmg \mid bmg_1 \vee bmg_2 \\
btg & ::= & thrg \mid \neg btg \mid btg_1 \vee btg_2 \\
bg & ::= & ling \mid \neg bg \mid bg_1 \vee bg_2
\end{array}
$$

Satisfaction of guards by intervals

Given a word $w \in A^+$ and $x, y \in dom(w)$,
let $\#B(w, x, y)$ denote the number of occurrences of letters in B at positions x to y inclusive.
Given group $G = \{h_1, \ldots, h_k, 0\}$, define a group value at position z:

$$G(w, z) = \begin{cases} c_j h_j, & \text{if } w[z] \in B_j \setminus (B_1 \cup \cdots \cup B_{j-1}), 1 \leq j \leq k, \\ 0, & \text{if } w[z] \notin (B_1 \cup \cdots \cup B_k) \end{cases}$$

Satisfaction of guards by intervals

Given a word $w \in A^+$ and $x, y \in dom(w)$,
let $\#B(w, x, y)$ denote the number of occurrences of letters in B at positions x to y inclusive.
Given group $G = \{h_1, \ldots, h_k, 0\}$, define a group value at position z:

$$G(w, z) = \begin{cases} c_j h_j, & \text{if } w[z] \in B_j \setminus (B_1 \cup \cdots \cup B_{j-1}), 1 \le j \le k, \\ 0, & \text{if } w[z] \notin (B_1 \cup \cdots \cup B_k) \end{cases}$$

- $w, [x, y] \models \Sigma_i c_i \#B_i \in R \bmod q$
  iff $\Sigma_i c_i \#B_i(w, x+1, y-1) \in R \bmod q$
- $w, [x, y] \models \Sigma_G(c_1 \#B_1, \ldots, c_k \#B_k) \in H$
  iff $\Sigma_{z=x+1}^{y-1} G(w, z) \in H$
- $w, [x, y] \models t \sim \#B \sim' u$
  iff $t \sim \#B(w, x+1, y-1) \sim' u$

In the temporal logic *PSL*, a semi-extended regular expression (intersections allowed) is a “triggering” interval constraint.

Logic *BLinTL*

$\phi ::= a \in A \mid \neg\phi \mid \phi \vee \phi \mid bg\,\mathsf{U}\,\phi \mid bg\,\mathsf{S}\,\phi$

$bg_1\,\mathsf{U}\,(bg_2\,\mathsf{U}\,\phi)$ is okay, cannot nest one U/S inside another.
Abbreviations: $B\,\mathsf{U}\,\phi$ for $(\#(A - B) = 0)\,\mathsf{U}\,\phi$. $\Box\phi = \neg\Diamond\neg\phi$, $\Diamond\phi = A\,\mathsf{U}\,\phi$, $\bigcirc\phi = \emptyset\,\mathsf{U}\,\phi$, $\ominus\phi = \emptyset\,\mathsf{S}\,\phi$.

$LTL[\Diamond, \diamondsuit, \bigcirc, \ominus]$, with counting in binary, is a subset.
PSL semi-extended regular expression triggers: $SRE \diamond\!\!\rightarrow \phi$.

Logic *BLinTL*

$\phi ::= a \in A \mid \neg\phi \mid \phi \vee \phi \mid bg\,\mathsf{U}\,\phi \mid bg\,\mathsf{S}\,\phi$

$bg_1\,\mathsf{U}\,(bg_2\,\mathsf{U}\,\phi)$ is okay, cannot nest one U/S inside another.
Abbreviations: $B\,\mathsf{U}\,\phi$ for $(\#(A - B) = 0)\,\mathsf{U}\,\phi$. $\Box\phi = \neg\Diamond\neg\phi$, $\Diamond\phi = A\,\mathsf{U}\,\phi$, $\bigcirc\phi = \emptyset\,\mathsf{U}\,\phi$, $\ominus\phi = \emptyset\,\mathsf{S}\,\phi$.

$LTL[\Diamond, \diamondsuit, \bigcirc, \ominus]$, with counting in binary, is a subset.
PSL semi-extended regular expression triggers: $SRE \diamond\!\!\rightarrow \phi$.

Given a word $w \in A^+$ and position $i \in dom(w)$:

- $w, i \models a$ iff $w[i] = a$
- $w, i \models bg\,\mathsf{U}\,\phi$ iff for some $j > i : w, [i, j] \models bg$ and $w, j \models \phi$
- $w, i \models bg\,\mathsf{S}\,\phi$ iff for some $j < i : w, [j, i] \models bg$ and $w, j \models \phi$
- $w, i \models SRE \diamond\!\!\rightarrow \phi$ iff
  for some $j \geq i : w[i..j] \in Lang(SRE)$ and $w, j \models \phi$

The same definitions would work for infinite words.

Sublogics

- Logic *BThTL*: modalities use only threshold constraints $btg\,\mathsf{U}\,\phi$ and $btg\,\mathsf{S}\,\phi$.
- Logic *InvTL*: modalities use only simple constraints $sg\,\mathsf{U}\,\phi$ and $sg\,\mathsf{S}\,\phi$.
- Logic *ModTL*: modalities use modulo constraints $bmg\,\mathsf{U}\,\phi$ and $bmg\,\mathsf{S}\,\phi$.
- Logic *InvModTL*: modalities use simple and modulo counting constraints $sg\,\mathsf{U}\,\phi$, $modg\,\mathsf{U}\,\phi$ and $sg\,\mathsf{S}\,\phi$, $modg\,\mathsf{S}\,\phi$.

Ideas also extend to group constraints.

Examples

- $\Box a \wedge ((\#a \equiv 0 \bmod 2) \,\mathsf{U}\, \neg \bigcirc \mathit{true})$ defines $(aa)^*$.

Examples

- $\Box a \wedge ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ defines $(aa)^*$.
- $(\#b \equiv 1 \bmod 3)\ \mathsf{U}\ ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ says that every word has $3n + 1$ occurrences of the letter b, for some $n \geq 0$, followed by an even number of occurrences of the letter a, excluding the last letter on the word.

Examples

- $\Box a \wedge ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ defines $(aa)^*$.
- $(\#b \equiv 1 \bmod 3)\ \mathsf{U}\ ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ says that every word has $3n + 1$ occurrences of the letter b, for some $n \geq 0$, followed by an even number of occurrences of the letter a, excluding the last letter on the word.
- $\mathit{Stair}_k = \Diamond(a \wedge (\#b = 0 \wedge \#a = k - 2)\ \mathsf{U}\ a)$ specifies k occurrences of the letter a without any intermediate occurrences of letter b. In *LTL* this would be written $\Diamond(a \wedge \neg a\ \mathsf{U}\ (a \wedge \neg a\ \mathsf{U}\ (\ldots a) \ldots))$, exponentially longer.

Examples

- $\Box a \wedge ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ defines $(aa)^*$.
- $(\#b \equiv 1 \bmod 3)\ \mathsf{U}\ ((\#a \equiv 0 \bmod 2)\ \mathsf{U}\ \neg \bigcirc \mathit{true})$ says that every word has $3n + 1$ occurrences of the letter b, for some $n \geq 0$, followed by an even number of occurrences of the letter a, excluding the last letter on the word.
- $\mathit{Stair}_k = \Diamond(a \wedge (\#b = 0 \wedge \#a = k - 2)\ \mathsf{U}\ a)$ specifies k occurrences of the letter a without any intermediate occurrences of letter b. In LTL this would be written $\Diamond(a \wedge \neg a\ \mathsf{U}\ (a \wedge \neg a\ \mathsf{U}\ (\ldots a) \ldots))$, exponentially longer.
- $U_2 = \Box(b \wedge (c\ \mathsf{U}\ b) \supset \Diamond\!\!\!-(a \wedge (c\ \mathsf{S}\ a)))$ defines a language in InvTL not definable in $\mathit{LTL}[\Diamond, \Diamond\!\!\!-]$:
  if a word has an occurrence of two b's without an a between them, then it must be preceded by an occurrence of two a's without a b between them.

Some known results

An *LTL* formula α is satisfiable if it holds in some word model σ at the beginning position 1.

Theorem (Ono and Nakamura'80; Sistla and Clarke'85)

Satisfiability for $LTL[U, S, \bigcirc, \ominus]$ *is in* PSPACE*; for* $LTL[\bigcirc, \Diamond]$ *is hard for* PSPACE*; for* $LTL[\Diamond]$ *satisfiability is in* NP*.*

Some known results

An LTL formula α is satisfiable if it holds in some word model σ at the beginning position 1.

Theorem (Ono and Nakamura'80; Sistla and Clarke'85)

Satisfiability for $LTL[U, S, \bigcirc, \ominus]$ is in PSPACE*; for $LTL[\bigcirc, \Diamond]$ is hard for* PSPACE*; for $LTL[\Diamond]$ satisfiability is in* NP*.*

Modelchecking question for these logics (same complexity):
A finite Kripke structure K describes all the word models and one has to check $K \models \alpha$.

Theorem (Sistla and Clarke'85; Etessami, Vardi, Wilke'02)

Modelchecking for $LTL[U, S, \bigcirc, \ominus]$ is in PSPACE*; for $LTL[\bigcirc, \Diamond]$ is hard for* PSPACE*; for $LTL[\Diamond]$ is in* NP*.*

Takeaway from this talk

- The early 1980s work of Pratt and of Harel, Kozen and Parikh, checked satisfiability of temporal and process logics using traditional filtration and tableau arguments. (Process logics combine point/interval temporal logics.)
- Soon after, Vardi and Wolper formulated arguments using automata over words, checking acceptance conditions for paths of temporal states, that some states are reached, perhaps repeatedly.
- Automata can do *threshold* and *modulo counting*, as well as finite *group* computation, tackled by the same algorithms.
- We added corresponding guards for U/S modalities retaining decidable satisfiability and modelchecking.

Upper bounds

- Constants encoded in binary: size of 1000 is 10.
- Size of set B is $|B|$.
  $|(\neg(\#\{b, c\} > 1) \wedge \#a = 17)\ \mathsf{U}\ a|$
  $= max(|\neg\#\{b, c\} > 1|,\ |\#a = 17|) + 1$
  $= max(2, \lceil \log_2 17 \rceil) + 1 = 6.$

Upper bounds

- Constants encoded in binary: size of 1000 is 10.
- Size of set B is $|B|$.
  $|(\neg(\#\{b, c\} > 1) \wedge \#a = 17)\ \mathsf{U}\ a|$
  $= max(|\neg\#\{b, c\} > 1|,\ |\#a = 17|) + 1$
  $= max(2, \lceil \log_2 17 \rceil) + 1 = 6$.

Theorem

Satisfiability of BLinTL is in ExpSpace.

- Consider that only a single modulus q occurs in α.
- (Least common multiple is polynomial in size of two numbers.)

Keeping track of values of guards

For initializing and updating guards:

- If g is $\Sigma_i c_i \# B_i \equiv r \bmod q$, then $g(0) := \Sigma_i c_i \# B_i \equiv 0 \bmod q$ and $g + c_j := \Sigma_i c_i \# B_i \equiv r + c_j \bmod q$.
- If g is $\Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h$, then the guard $g(0) := \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = 0$ and the guard $g + h' := \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h + h'$.
- If g is $\# B \sim v$ and $a \in A$, then $g - a := \# B \sim v - 1$ if $a \in B$, and g otherwise.

Keeping track of values of guards

For initializing and updating guards:

- If g is $\Sigma_i c_i \# B_i \equiv r \bmod q$, then $g(0) := \Sigma_i c_i \# B_i \equiv 0 \bmod q$ and $g + c_j := \Sigma_i c_i \# B_i \equiv r + c_j \bmod q$.
- If g is $\Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h$, then the guard $g(0) := \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = 0$ and the guard $g + h' := \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h + h'$.
- If g is $\# B \sim v$ and $a \in A$, then $g - a := \# B \sim v - 1$ if $a \in B$, and g otherwise.

For guard g we introduce a derived modality:

Now $g = g \,\mathsf{S}\, (\neg \ominus \mathit{true})$ gives the current value of a guard evaluated from the first position of the word.

Global counter validities

$$(\Sigma_i c_i \#B_i \equiv r \bmod q)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [q]} (\mathit{Now}\ \Sigma_i c_i \#B_i \equiv r_0 \bmod q) \wedge \Diamond(\phi \wedge (\mathit{Now}\ \Sigma_i c_i \#B_i \equiv r_0 + r \bmod q))$$

$$(\#B < u)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,\phi$$

$$(\#B = u - 1)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,(\phi \wedge (\mathit{Now}\ \#B \equiv r_0 - 1 \bmod u))$$

$$(t \leq \#B < u)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 + t \bmod u)\mathsf{U}\ ((\mathit{Now}\ \#B \equiv r_0 + t \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,\phi)$$

Global counter validities

$$(\Sigma_i c_i \#B_i \equiv r \bmod q)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [q]} (\mathit{Now}\ \Sigma_i c_i \#B_i \equiv r_0 \bmod q) \wedge \Diamond(\phi \wedge (\mathit{Now}\ \Sigma_i c_i \#B_i \equiv r_0 + r \bmod q))$$

$$(\#B < u)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,\phi$$

$$(\#B = u - 1)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,(\phi \wedge (\mathit{Now}\ \#B \equiv r_0 - 1 \bmod u))$$

$$(t \leq \#B < u)\,\mathsf{U}\,\phi \Leftrightarrow \bigvee_{r_0 \in [u]} (\mathit{Now}\ \#B \equiv r_0 \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 + t \bmod u)\mathsf{U}\ ((\mathit{Now}\ \#B \equiv r_0 + t \bmod u) \wedge \neg(\mathit{Now}\ \#B \equiv r_0 \bmod u)\,\mathsf{U}\,\phi)$$

- In each case, only one right hand disjunct can hold.
- The right hand disjuncts do not have threshold constraints: threshold counting is reduced to modulo counting.

Fischer-Ladner subformula closure

Fix formula α_0 whose satisfiability we want to check.

- α_0 is in the closure of α_0.
- If ϕ is in the closure, $\neg\phi$ is in the closure.
  We identify $\neg\neg\phi$ with ϕ.
- If $\phi \vee \psi$, $\phi \,\mathsf{U}\, \psi$ and $\phi \,\mathsf{S}\, \psi$ are in the closure, so are ϕ, ψ.
- For an interval-guarded formula in our syntax, we put into the closure an appropriate formula from the global counter validities on the previous slide. (Details on the next slide.)
- (For a *PSL* expression-triggered formula $SRE \diamond\!\!\rightarrow \phi$ the closure has formulas $DE \diamond\!\!\rightarrow \phi$, for all "derivative" expressions DE of SRE.)

Global counters as subformulae

- The closure of a set with $(\Sigma_i c_i \# B_i \in R \bmod q)\ \mathsf{U}\ \phi$ includes: $\Diamond(\phi \wedge \mathit{Now}\ \Sigma_i c_i \# B_i \equiv r \bmod q)$ and $\mathit{Now}\ \Sigma_i c_i \# B_i \equiv r \bmod q$, for every $0 \leq r < q$.
- The closure of a set with $(\Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) \in H)\ \mathsf{U}\ \phi$ includes: $\Diamond(\phi \wedge \mathit{Now}\ \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h)$ and $\mathit{Now}\ \Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h$, for every $h \in G$.
- The closure of a set with $(t \leq \# B < u)\ \mathsf{U}\ \phi$ includes: $(\neg(\mathit{Now}\ \# B \equiv r \bmod u)\mathsf{U}$ $((\mathit{Now}\ \# B \equiv r \bmod u) \wedge \Diamond(\phi \wedge \mathit{Now}\ \# B \equiv s \bmod u))$, for every r, s in $\{0, \ldots, q-1\}$.

Size of closure and atoms

- Once a formula $\delta \equiv r \bmod q$ enters the closure of the given α_0, the entire set $\{\delta \equiv r \bmod q \mid 0 \leq r < q\}$ has to be included in the closure of α_0. This is exponential in q and hence exponential in the size of α_0.
- For a threshold constraint formula the closure is $O(2^{|\alpha_0|^2})$.
- For every constraint, only one of the global counter formulae with modulus value $0 \leq r < q$, or with group element value $h \in G$, can hold in a state.
- Hence the number of states grows only exponentially with the size of α_0, even though the modulo and group counting constants are represented in binary.

The formula automaton

- There is a finite formula automaton M_α of size exponential in α which accepts exactly the language of word models of α.
- Each of its states can be represented in polynomial space.
- On the next slide we define when there is a transition from s_1 to s_2.
- Without loss of generality all subalphabets B_i mentioned in an alphabetic condition on the next slide are disjoint from each other.

The transition relation

1. If g is $\Sigma_i c_i \# B_i \equiv r \bmod q$ and if $a \in B_j$ for some j is in s_2, then:
   - $g \,\mathsf{U}\, \phi$ in s_2 implies $(g + c_j) \,\mathsf{U}\, \phi$ in s_1;
   - $g \,\mathsf{U}\, \phi$ in s_1 implies $(g - c_j) \,\mathsf{U}\, \phi$ in s_2.
2. If g is $\Sigma_G(c_1 \# B_1, \ldots, c_k \# B_k) = h$ and if $a \in B_j$ for some j is in s_2, then:
   - $g \,\mathsf{U}\, \phi$ in s_2 implies $(g + c_j h_j) \,\mathsf{U}\, \phi$ in s_1;
   - $g \,\mathsf{U}\, \phi$ in s_1 implies $(g - c_j h_j) \,\mathsf{U}\, \phi$ in s_2.
3. If g is a threshold constraint $\# B = 0$ and if $a \in B$ is in s_1, then $g \,\mathsf{U}\, \phi$ in s_1 implies ϕ in s_2.
4. In each case above, if the alphabetic precondition is not satisfied, depending on whether it was assumed to be in s_2 or s_1, then $g \,\mathsf{U}\, \phi$ is required to be in s_1 or s_2 respectively.

Extends to *PSL* triggers using “derivative” expressions.

Decision procedure for satisfiability

Theorem

The satisfiability problem for BLinTL is in ExpSpace.

- The formula automaton has exponentially many states.
- An accepting path in the automaton may require going through an entire range of global counter values, and with several such counters operating.
- The accepting path has to be guessed, written down and verified. This can be done in ExpSpace.
- The sublogic *BThTL* can define counters upto exponential values, this makes the lower bound ExpSpace.

Decision procedure for satisfiability

Theorem

The satisfiability problem for BLinTL is in ExpSpace.

- The formula automaton has exponentially many states.
- An accepting path in the automaton may require going through an entire range of global counter values, and with several such counters operating.
- The accepting path has to be guessed, written down and verified. This can be done in ExpSpace.
- The sublogic *BThTL* can define counters upto exponential values, this makes the lower bound ExpSpace.

Ideas extend to triggers in *PSL* (decidable in ExpSpace).
Not to infinite groups such as $\mathbb{Z}$: can code counter machines.

Decision procedure for model checking

Corollary

The model checking problem for BLinTL is PSPACE *in the size of the model and* EXPSPACE *in the size of the formula.*

- Let α_0 be a formula and K a Kripke structure.
- The theorem shows that for formula $\neg\alpha_0$ there is an exponential size formula automaton $M(\neg\alpha_0)$.
- Verifying $K \models \alpha_0$ is equivalent to checking whether the intersection of the languages corresponding to K and $M(\neg\alpha_0)$ is nonempty.
- Uses space logarithmic in the size of both the models. Since $M(\neg\alpha_0)$ is exponentially larger than α_0 we get the upper bounds using Savitch's theorem.
- Lower bounds already known from Counting *LTL*.

Philosophical to industrial (Vardi, ICLA'09)

- Our ideas extend to semi-extended regular expression triggers in the industry standard temporal logic *PSL*.
- Linear arithmetic is widely used in combination with SAT-solvers, which practical implementations may use.
- We constructed the usual nondeterministic finite automata studied in formal language theory. There is a different approach using alternating finite automata, which can be "compiled" when required into nondeterministic automata.

Branching time

- Our approach extends to logic *CTL* with interval constraints.
- The formula automaton constructed uses states derived from the Fischer-Ladner closure as above, but works on *trees* instead of words.
- A transition relation connects a state to several states, the arity is determined by the number of existential U/S requirements in a state.
- We can prove that satisfiability is in 2ExpTime.
- Laroussinie, Meyer and Petonnet obtained the same upper bound by an exponential translation to ordinary *CTL* (satisfiability in ExpTime), also a matching lower bound.

First-order logic and two-variable logics

Theorem (Kamp'68; Etessami, Vardi and Wilke'02)

On word models, $LTL[\bigcirc, U]$ is as expressive as first-order logic. $LTL[\Diamond, \Diamond\!\!\!-, \bigcirc, \ominus]$ has the same expressiveness as FO^2 logic.

- Logic FO^2 was studied by Mortimer in mid-1970s. Grädel, Otto and Rosen and Pacholski, Szwast and Tendera pinned down its complexity in late 1990s.
- Formulas of logic $FO^2\text{MOD}[<]$ (Straubing and Thérien) and $FO^2[Bet]$ (with Krebs, Pandya and Straubing) define subclasses of regular languages.
- Our logic *BLinTL* defines a bigger (still regular) subclass encompassing both and is in $FO^2\text{MOD}[Bet]$.

Open: ranker structure, see deterministic logics in part 1 of talk.
Open: algorithmic characterization of expressiveness.