Cost Linear Temporal Logic for Verification

Maximilien Colange, Dimitri Racordon, Didier Buchs

University of Geneva

January 30, 2015

Quantitative Verification

Qualitative Verification

Yes-No Question

Quantitative Verification

Question has an answer in a domain D

Quantitative Verification

Qualitative Verification

Yes-No Question

Quantitative Verification

Question has an answer in a domain D

- D continuous (time, probability) $\Rightarrow$ dedicated models
  timed, hybrid, stochastic ...

Quantitative Verification

Qualitative Verification

Yes-No Question

Quantitative Verification

Question has an answer in a domain D

- D continuous (time, probability) $\Rightarrow$ dedicated models timed, hybrid, stochastic ...
- D discrete $\Rightarrow$ same models as qualitative setting

Quantitative Verification

Qualitative Verification

Yes-No Question

Quantitative Verification

Question has an answer in a domain D

- D continuous (time, probability) $\Rightarrow$ dedicated models
  timed, hybrid, stochastic ...
- D discrete $\Rightarrow$ same models as qualitative setting
  - yes, but inefficient

Quantitative Verification

Qualitative Verification

Yes-No Question

Quantitative Verification

Question has an answer in a domain D

- D continuous (time, probability) $\Rightarrow$ dedicated models timed, hybrid, stochastic ...
- D discrete $\Rightarrow$ same models as qualitative setting
  - yes, but inefficient

We focus on *discrete quantitative*

Motivating Example

Dining philosophers (ressource sharing model)

- N philosophers, N forks
- philo either thinking, or eating
- 2 forks needed to eat
- Starvation = does a philo not eat forever?

Motivating Example

Dining philosophers (ressource sharing model)

- N philosophers, N forks
- philo either thinking, or eating
- 2 forks needed to eat
- Starvation $=$ does a philo not eat forever?

In real life, starvation occurs in finite time

Motivating Example

Dining philosophers (ressource sharing model)

- N philosophers, N forks
- philo either thinking, or eating
- 2 forks needed to eat
- Starvation = does a philo not eat forever?

In real life, starvation occurs in finite time

- How long a philo thinks? (discrete time)
- Bounds on $\Rightarrow$ max/min thinking time (per philo, globally ...)

Motivating Example

Instrumentation: How to measure thinking (logical) time?

add a variable timer_p

```
if (philo p thinks) ++timer_p;
if (philo p eats) timer_p = 0;
```

Qualitative Verification of the instrumented model.

Motivating Example

Instrumentation: How to measure thinking (logical) time?

add a variable timer_p

```
if (philo p thinks) ++timer_p;
if (philo p eats) timer_p = 0;
```

Qualitative Verification of the instrumented model.

Numerous Drawbacks

- model modification = strong semantical risk
- translate quantitative to qualitative: values known a priori

Objective

Verification principle: model and property are independent
Quantitative measure should remain in the logic.

Proposal

Use a quantitative logic towards “discrete quantitative” verification

- Linear Temporal Logic very popular
- extend LTL with counting

LTL$^{\leq}$ [Kuperberg and Boom, 2012]

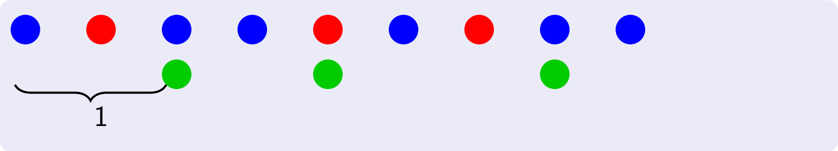
LTL$^{\leq}$ = LTL + a **U** operator that counts.

● $\mathbf{U}^{\leq}$ ●: counts $\neg$● until ●

LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

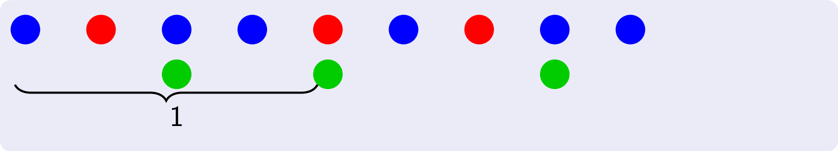
$\bullet$ $\mathbf{U}^{\leq}$ $\bullet$: counts $\neg\bullet$ until $\bullet$



LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

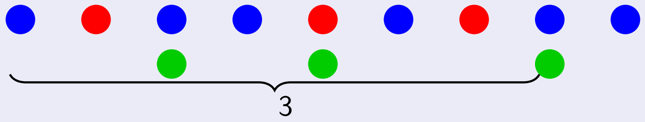
● $\mathbf{U}^{\leq}$ ●: counts $\neg$● until ●



LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

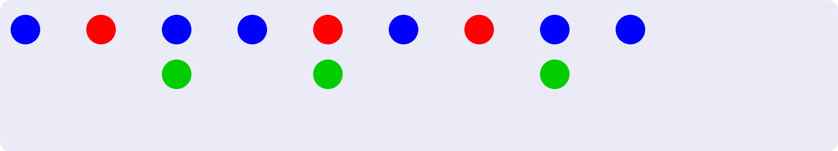
$\mathbf{U}^{\leq}$: counts $\neg$ until



LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

● $\mathbf{U}^{\leq}$ ●: counts $\neg$● until ●

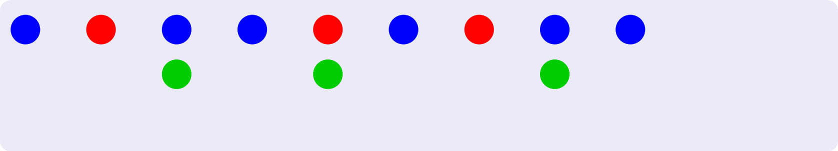


- find a n that dominates # of $\neg$●

LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

$\mathbf{U}^{\leq}$: counts $\neg$ until

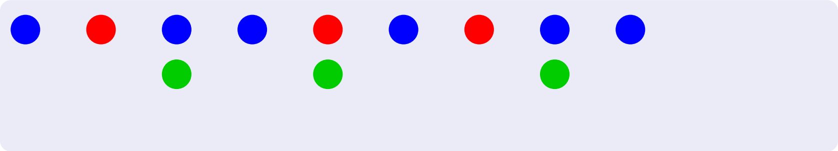


- find a n that dominates # of $\neg$
- $\{$possible $n\}$ $\uparrow$-closed

$\mathsf{LTL}^{\leq}$ [Kuperberg and Boom, 2012]

$\mathsf{LTL}^{\leq}$ = LTL + a **U** operator that counts.

● $\mathbf{U}^{\leq}$ ●: counts ¬● until ●

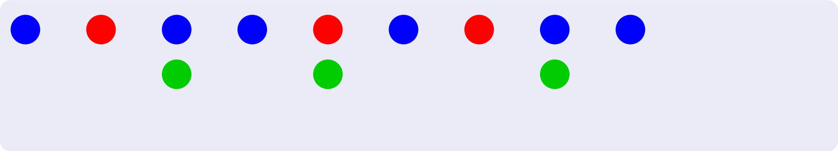


- find a n that dominates # of ¬●
- {possible n} ↑-closed
- same n for all occurrences of $\mathbf{U}^{\leq}$ in the formula

LTL$^{\leq}$ [Kuperberg and Boom, 2012]

LTL$^{\leq}$ = LTL + a **U** operator that counts.

● $\mathbf{U}^{\leq}$ ●: counts $\neg$● until ●



- find a n that dominates # of $\neg$●
- $\{$possible $n\}$ $\uparrow$-closed
- same n for all occurrences of $\mathbf{U}^{\leq}$ in the formula
- $[\![\phi]\!]_{\leq} = \inf\{\text{possible } n\}$

$\mathsf{LTL}^{>}$ [Kuperberg and Boom, 2012]

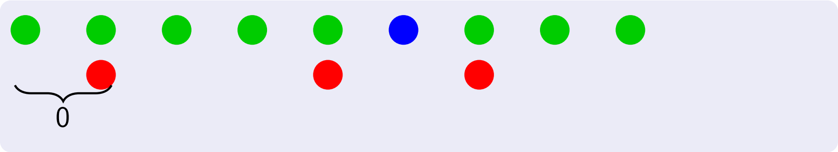
$\mathsf{LTL}^{>} = \mathsf{LTL} +$ a $\mathbf{R}$ operator that counts.

$\mathbf{R}^{>}$: counts while

$\mathsf{LTL}^{>}$ [Kuperberg and Boom, 2012]

$\mathsf{LTL}^{>}$ = LTL + a **R** operator that counts.

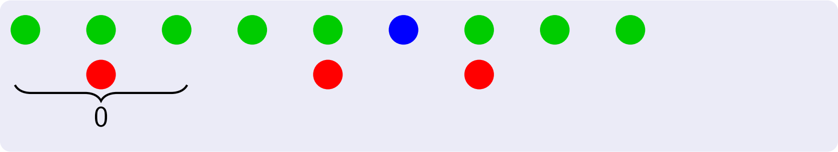
🔴 $\mathbf{R}^{>}$ 🟢: counts 🔴 while 🟢



$\mathrm{LTL}^{>}$ [Kuperberg and Boom, 2012]

$\mathrm{LTL}^{>} = \mathrm{LTL} +$ a **R** operator that counts.

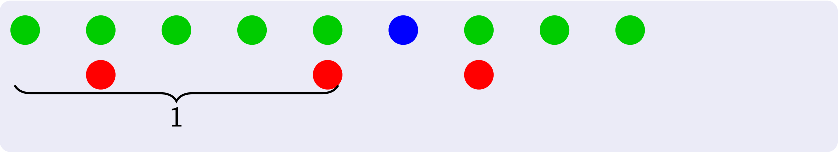
$\mathbf{R}^{>}$: counts while



$\mathsf{LTL}^{>}$ [Kuperberg and Boom, 2012]

$\mathsf{LTL}^{>} = \mathsf{LTL} +$ a **R** operator that counts.

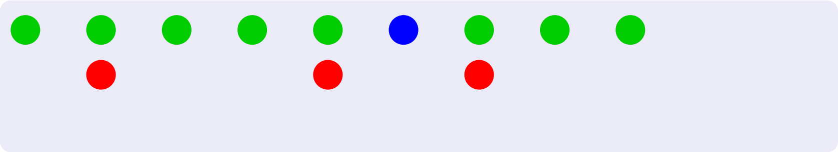
$\mathbf{R}^{>}$: counts while



LTL$^{>}$ [Kuperberg and Boom, 2012]

LTL$^{>}$ $=$ LTL $+$ a **R** operator that counts.

$\mathbf{R}^{>}$: counts while

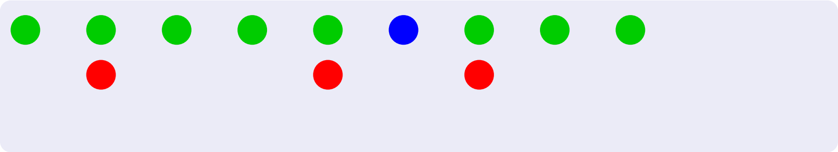


- find a n dominated by $\#$ of

LTL$^{>}$ [Kuperberg and Boom, 2012]

LTL$^{>}$ $=$ LTL $+$ a **R** operator that counts.

R$^{>}$: counts while

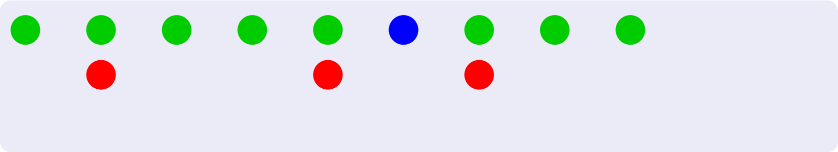


- find a n dominated by $\#$ of
- $\{$possible $n\}$ $\downarrow$-closed

$\mathrm{LTL}^{>}$ [Kuperberg and Boom, 2012]

$\mathrm{LTL}^{>} = \mathrm{LTL} + $ a **R** operator that counts.

$\mathbf{R}^{>}$: counts while

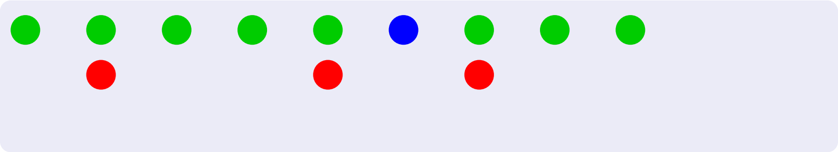


- find a n dominated by $\#$ of
- $\{$possible $n\}$ $\downarrow$-closed
- same n for all occurrences of $\mathbf{R}^{>}$ in the formula

$\mathrm{LTL}^{>}$ [Kuperberg and Boom, 2012]

$\mathrm{LTL}^{>} = \mathrm{LTL} +$ a **R** operator that counts.

R$^{>}$: counts while

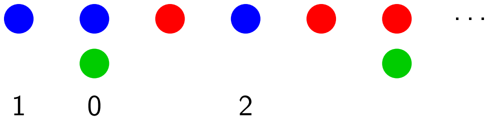


- find a n dominated by $\#$ of
- $\{$possible $n\}$ $\downarrow$-closed
- same n for all occurrences of **R**$^{>}$ in the formula
- $[\![\phi]\!]_{>} = \sup\{$possible $n\}$

Example

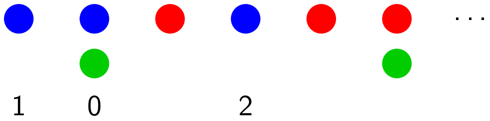
$\mathbf{G}\,(\textcolor{blue}{\bullet} \implies (\perp\ \mathbf{U}^{\leq}\ \textcolor{green}{\bullet}))$

counts the minimal distance between any $\textcolor{blue}{\bullet}$ and the next $\textcolor{green}{\bullet}$



Example

$\mathbf{G}\,(\quad \Longrightarrow (\perp\ \mathbf{U}^{\leq} \quad))$
counts the minimal distance between any and the next



$(u, n) \models \phi$ iff n dominates every $\mathbf{U}^{\leq}$

- here $(u, n) \models \phi$ iff $n \geq 2$

$[\![\phi]\!]_{\leq}(u) = \inf\{\text{possible } n\} = 2$

Negation and Duality

in $LTL^{\leq}$ and $LTL^{>}$, cost operators cannot be negated

Remark

$$(u,n) \models a \, \mathbf{U}^{\leq} \, b \quad \iff \quad (u,n) \not\models \neg a \, \mathbf{R}^{>} \, \neg b$$

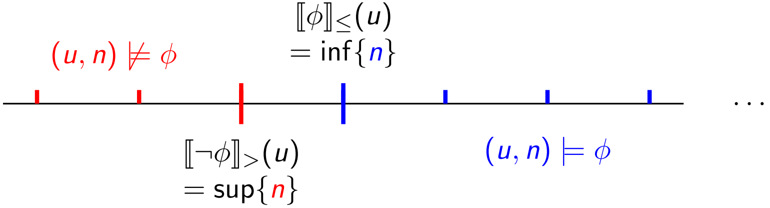
Duality through negation

$$LTL^{\leq} \underset{\neg}{\overset{\neg}{\rightleftharpoons}} LTL^{>}$$

and push negations to leaves (Negative Normal Form)

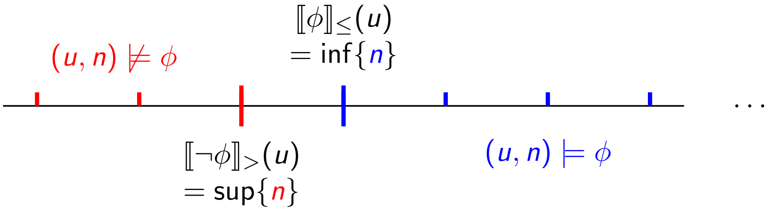
Negation and Semantics

$$(u, n) \models \phi \iff (u, n) \not\models \neg\phi$$



Negation and Semantics

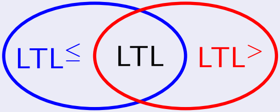
$$(u, n) \models \phi \iff (u, n) \not\models \neg\phi$$



- semantically, $\neg$ is ± 1

And LTL?

Syntactically



Semantically ($\phi \in LTL$)

$$u \vdash \phi \iff \forall n \in \mathbb{N}.(u, n) \models \phi$$

$LTL^{\leq}$	LTL	$LTL^{>}$
$[\![\phi]\!]_{\leq}(u) = 0$	$u \vdash \phi$	$[\![\phi]\!]_{>}(u) = +\infty$
$[\![\phi]\!]_{\leq}(u) = +\infty$	$u \not\vdash \phi$	$[\![\phi]\!]_{>}(u) = 0$

$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ model-checking

$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ more expressive than LTL
New problems arise

- $(u, n) \models \phi$? for a given $n \in \mathbb{N}$
  - $\exists u$
  - $\forall u$

$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ model-checking

$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ more expressive than LTL
New problems arise

- $(u, n) \models \phi$? for a given $n \in \mathbb{N}$
  - $\exists u$
  - $\forall u$
- is $[\![\phi]\!]_{\leq}$ bounded over a given language L?
  - decidable when L regular [Bojańczyk and Colcombet, 2006]

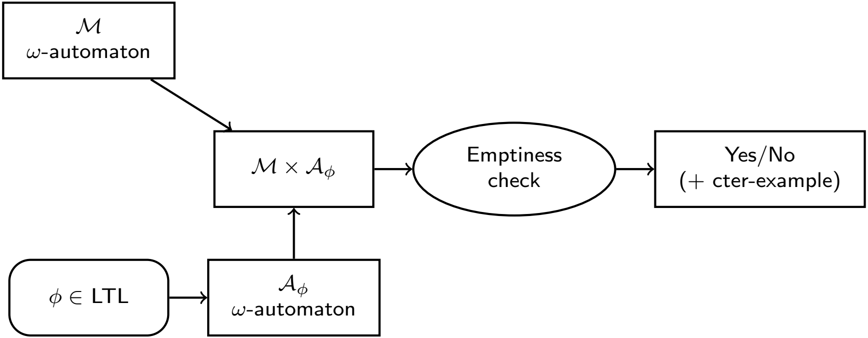
$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ model-checking

$\mathrm{LTL}^{\leq}$ and $\mathrm{LTL}^{>}$ more expressive than LTL

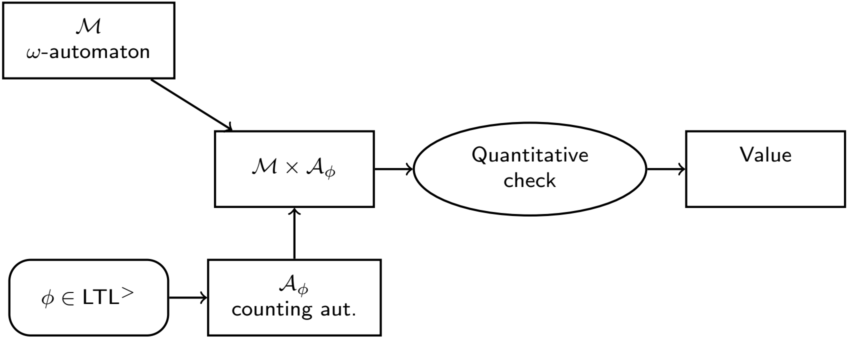
New problems arise

- $(u, n) \models \phi$? for a given $n \in \mathbb{N}$
  - $\exists u$
  - $\forall u$
- is $[\![\phi]\!]_{\leq}$ bounded over a given language L?
  - decidable when L regular [Bojańczyk and Colcombet, 2006]
- actual values of bounds over a given set L
  - $\sup_L [\![\phi]\!]_{\leq}$ ($\mathrm{LTL}^{>}$ seems appropriate)
  - $\inf_L [\![\phi]\!]_{\leq}$ ($\mathrm{LTL}^{\leq}$ seems appropriate)

Following the Automata Approach: LTL



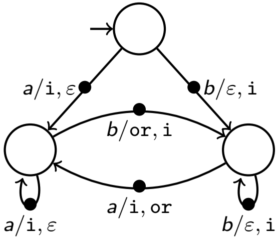
Following the Automata Approach: $\mathsf{LTL}^{>}$



Cost Register Automata

[Bojańczyk and Colcombet, 2006]

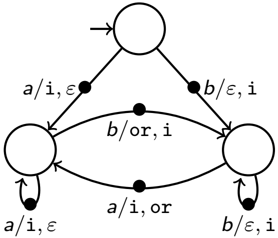
Transition-Based Generalized Büchi Automaton
+ finite set of counters
actions on counter: observe (o), increment (i), reset (r), ε



Cost Register Automata

[Bojańczyk and Colcombet, 2006]

Transition-Based Generalized Büchi Automaton
$+$ finite set of counters
actions on counter: observe ($\texttt{o}$), increment ($\texttt{i}$), reset ($\texttt{r}$), ε



2 counters

- 1st one counts consecutive a's
- 2nd one counts consecutive b's

$$Val(\rho) = \min\{\text{observed values}\}$$
$$= \min \# \text{ consecutive occurrences of the same letter}$$

Cost Register Automata Semantics

$>$-automata

- $\mathrm{Val}_>(\rho) = \inf\{\text{observed counter values}\}$
- $[\![\mathcal{A}]\!]_>(u) = \sup_{\rho \text{ acc. run on } u} \mathrm{Val}_>(\rho)$

actions: `i`, `or`, `r`, ε

Cost Register Automata Semantics

$>$-automata

- $\mathrm{Val}_{>}(\rho) = \mathsf{inf}\{\text{observed counter values}\}$
- $[\![\mathcal{A}]\!]_{>}(u) = \mathsf{sup}_{\rho \text{ acc. run on } u} \mathrm{Val}_{>}(\rho)$

actions: `i`, `or`, `r`, ε

$\leq$-automata

- $\mathrm{Val}_{\leq}(\rho) = \mathsf{sup}\{\text{observed counter values}\}$
- $[\![\mathcal{A}]\!]_{\leq}(u) = \mathsf{inf}_{\rho \text{ acc. run on } u} \mathrm{Val}_{\leq}(\rho)$

actions: `io`, `r`, ε

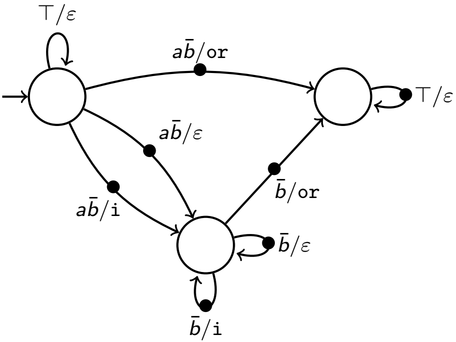
From LTL$^{>}$ to $>$-automata

Translation idea [Kuperberg, 2012]

- similar to LTL $\rightarrow$ Büchi translation
- accepting conditions on the transitions
- one counter for each $\mathbf{R}^{>}$
  - counts the occurrences of lhs of $\mathbf{R}^{>}$, as expected

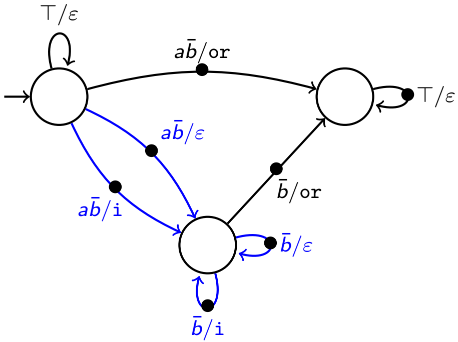
Automaton for $\phi = \mathbf{G}(a \implies \bot \mathbf{U}^{\leq} b)$

$[\![\phi]\!]_{\leq}(u) =$ max distance between any a and the next b



Automaton for $\phi = \mathbf{G}(a \implies \bot\mathbf{U}^{\leq} b)$

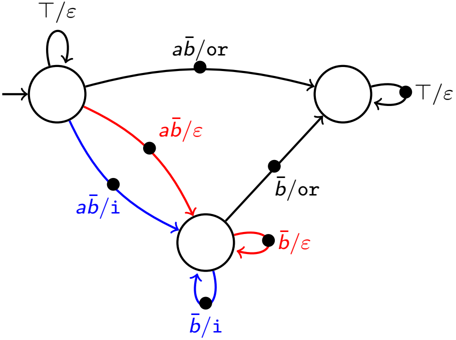
$[\![\phi]\!]_{\leq}(u) =$ max distance between any a and the next b



- non-determinism

Automaton for $\phi = \mathbf{G}(a \implies \perp \mathbf{U}^{\leq} b)$

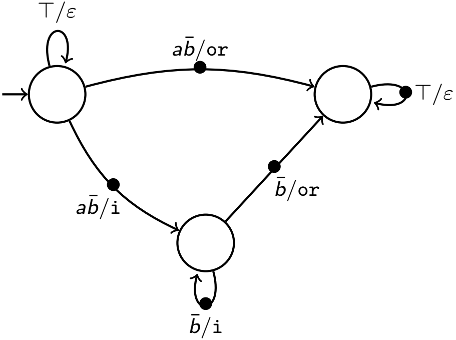
$[\![\phi]\!]_{\leq}(u) = \max$ distance between any a and the next b



- non-determinism
- run with largest value wins: useless transitions

Automaton for $\phi = \mathbf{G}(a \implies \bot\mathbf{U}^{\leq} b)$

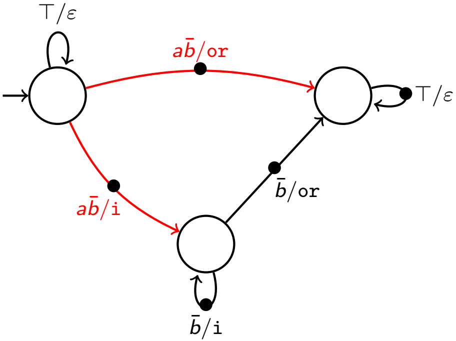
$[\![\phi]\!]_{\leq}(u) = \max$ distance between any a and the next b



- non-determinism
- run with largest value wins: useless transitions
- safe to remove useless transitions

Automaton for $\phi = \mathbf{G}(a \implies \bot \mathbf{U}^{\leq} b)$

$[\![\phi]\!]_{\leq}(u) = \max$ distance between any a and the next b

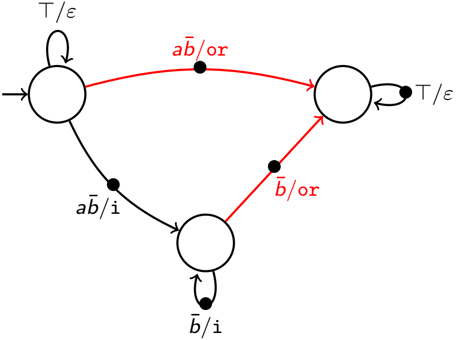


- non-determinism
- run with largest value wins: useless transitions
- safe to remove useless transitions

- guess the a for which the max is reached

Automaton for $\phi = \mathbf{G}(a \implies \bot \mathbf{U}^{\leq} b)$

$[\![\phi]\!]_{\leq}(u) =$ max distance between any a and the next b

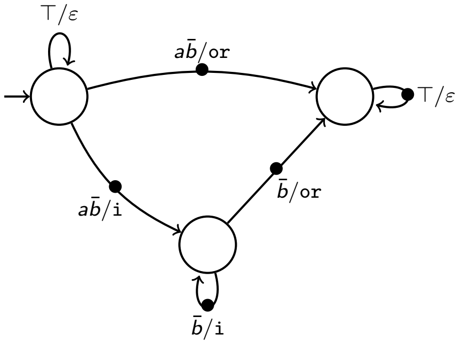


- non-determinism
- run with largest value wins: useless transitions
- safe to remove useless transitions

- guess the a for which the max is reached
- guess: or just before b

Automaton for $\phi = \mathbf{G}(a \implies \bot\mathbf{U}^{\leq} b)$

$[\![\phi]\!]_{\leq}(u) =$ max distance between any a and the next b



- non-determinism
- run with largest value wins: useless transitions
- safe to remove useless transitions

- guess the a for which the max is reached
- guess: `or` just before b
- wrong guesses yield runs with smaller values, eliminated by sup in $[\![\mathcal{A}]\!]_{>}$

Compute the Upper Bound of a $>$-Automata

Several sub-problems.

Boundedness

Is $\sup[\![\mathcal{A}]\!]_>$ finite?

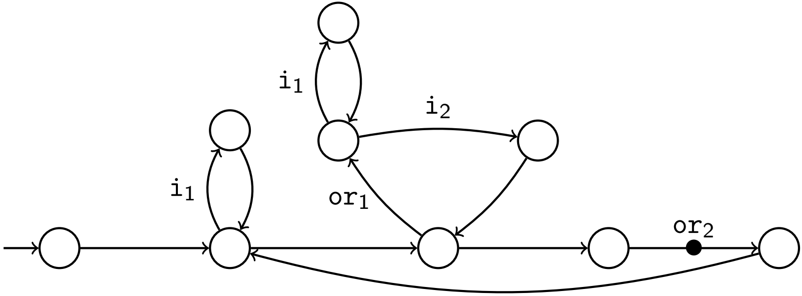
Exact value

If finite, value of $\sup[\![\mathcal{A}]\!]_>$?

Boundedness [Colcombet, 2009]

Property

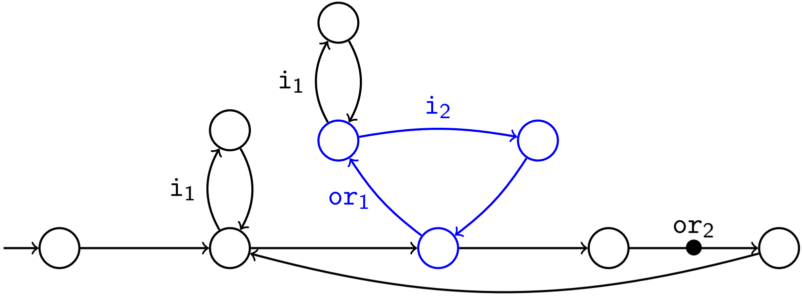
sup$[\![\mathcal{A}]\!]_{>} = \infty$ iff $\exists \rho$ acc. run s.t.
every $\texttt{or}_\gamma$ is preceded by a cycle that increments γ and never $\texttt{or}$ it (γ-cycle).



Boundedness [Colcombet, 2009]

Property

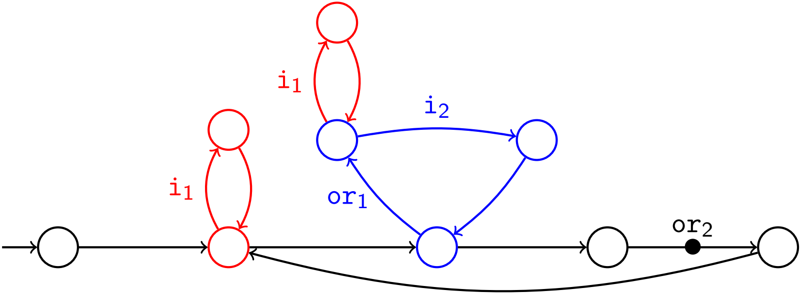
$\sup[\![\mathcal{A}]\!]_{>} = \infty$ iff $\exists \rho$ acc. run s.t.
every $\mathtt{or}_\gamma$ is preceded by a cycle that increments γ and never $\mathtt{or}$ it (γ-cycle).



Boundedness [Colcombet, 2009]

Property

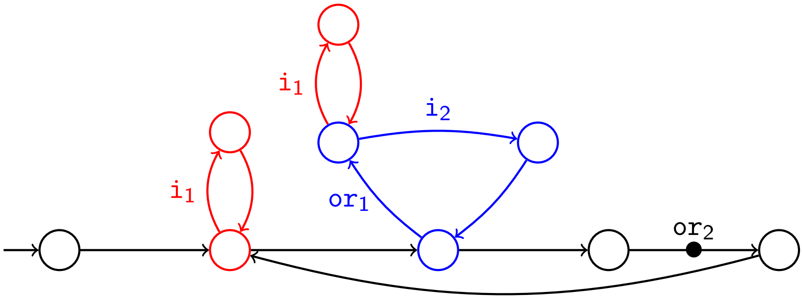
sup$[\![\mathcal{A}]\!]_{>} = \infty$ iff $\exists \rho$ acc. run s.t.
every $\mathtt{or}_\gamma$ is preceded by a cycle that increments γ and never or it (γ-cycle).



Boundedness [Colcombet, 2009]

Property

sup$[\![\mathcal{A}]\!]_> = \infty$ iff $\exists \rho$ acc. run s.t.
every $\texttt{or}_\gamma$ is preceded by a cycle that increments γ and never $\texttt{or}$ it (γ-cycle).



Upper bound

if bounded, sup$[\![\mathcal{A}]\!]_> \leq |Q_{\mathcal{A}}|$

Back to LTL

Büchi Emptiness Check

$u \in L(\mathcal{A})$ iff $\exists$ acc. run on u
possible early answer

Bound Computation

$\text{sup}[\![A]\!]_{>} = \text{sup}_{\text{all acc. runs } \rho} \mathrm{Val}_{>}(\rho)$
no early break is possible
but once a candidate n is found, all values $< n$ are ruled out

Back to LTL

Büchi Emptiness Check

$u \in L(\mathcal{A})$ iff $\exists$ acc. run on u
possible early answer

Bound Computation

$\sup[\![A]\!]_{>} = \sup_{\text{all acc. runs } \rho} \mathrm{Val}_{>}(\rho)$
no early break is possible
but once a candidate n is found, all values $< n$ are ruled out

Problem: remove runs of value $< n$ in $\mathcal{A}$

Back to LTL

Problem

INPUT: $\phi \in \mathrm{LTL}^{>}, n \in \mathbb{N}$

OUTPUT: $\phi[n] \in \mathrm{LTL}$ s.t. $u \vdash \phi[n] \iff [\![\phi]\!]_{>}(u) \geq n$

n is "hardcoded" in $\phi[n]$

Back to LTL

Problem

INPUT: $\phi \in \mathrm{LTL}^{>}, n \in \mathbb{N}$

OUTPUT: $\phi[n] \in \mathrm{LTL}$ s.t. $u \vdash \phi[n] \iff [\![\phi]\!]_{>}(u) \geq n$

- for $\phi, \psi \in \mathrm{LTL}$
  - $(\phi\ \mathbf{R}^{>}\ \psi)[0] \equiv \phi\ \mathbf{R}\ \psi$
  - $(\phi\ \mathbf{R}^{>}\ \psi)[n] \equiv (\phi \wedge \mathbf{X}\ (\phi\ \mathbf{R}^{>}\ \psi)[n-1])\ \mathbf{R}\ \psi$
- otherwise $(\phi \bowtie \psi)[n] = \phi[n] \bowtie \psi[n]$

n is "hardcoded" in $\phi[n]$

Back to LTL

Problem

INPUT: $\phi \in \mathrm{LTL}^{>}, n \in \mathbb{N}$
OUTPUT: $\phi[n] \in \mathrm{LTL}$ s.t. $u \vdash \phi[n] \iff [\![\phi]\!]_{>}(u) \geq n$

- for $\phi, \psi \in \mathrm{LTL}$
  - $(\phi\ \mathbf{R}^{>}\ \psi)[0] \equiv \phi\ \mathbf{R}\ \psi$
  - $(\phi\ \mathbf{R}^{>}\ \psi)[n] \equiv (\phi \wedge \mathbf{X}\ (\phi\ \mathbf{R}^{>}\ \psi)[n-1])\ \mathbf{R}\ \psi$
- otherwise $(\phi \bowtie \psi)[n] = \phi[n] \bowtie \psi[n]$

$\phi = a\ \mathbf{R}^{>}\ b,\ n = 1$

$(a \wedge \mathbf{X}\ (a\ \mathbf{R}\ b))\ \mathbf{R}\ b$

n is "hardcoded" in $\phi[n]$

LTL to remove runs

Recall

$[\![\phi_1 \wedge \phi_2]\!]_> = \sup\{n \text{ possible for } \phi_1 \text{ AND } \phi_2\} = \min([\![\phi_1]\!]_>, [\![\phi_2]\!]_>)$

LTL to remove runs

Recall

$[\![\phi_1 \wedge \phi_2]\!]_> = \sup\{n \text{ possible for } \phi_1 \text{ AND } \phi_2\} = \min([\![\phi_1]\!]_>, [\![\phi_2]\!]_>)$

$u \vdash \phi[n] \iff [\![\phi]\!]_>(u) \geq n$

$$[\![\phi[n]]\!]_>(u) = \begin{cases} +\infty & \text{if } [\![\phi]\!]_>(u) \geq n \\ 0 & \text{otherwise} \end{cases}$$

LTL to remove runs

Recall

$[\![\phi_1 \wedge \phi_2]\!]_> = \sup\{n \text{ possible for } \phi_1 \text{ AND } \phi_2\} = \min([\![\phi_1]\!]_>, [\![\phi_2]\!]_>)$

$u \vdash \phi[n] \iff [\![\phi]\!]_>(u) \geq n$

$$[\![\phi[n]]\!]_>(u) = \begin{cases} +\infty & \text{if } [\![\phi]\!]_>(u) \geq n \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\phi \wedge \phi[n]]\!]_>(u) = \begin{cases} [\![\phi]\!]_>(u) & \text{if } [\![\phi]\!]_>(u) \geq n \\ 0 & \text{otherwise} \end{cases}$$

$\phi[n]$ refines ϕ by forbidding words of value $< n$

LTL to remove runs

Recall

$[\![\phi_1 \wedge \phi_2]\!]_> = \sup\{n \text{ possible for } \phi_1 \text{ AND } \phi_2\} = \min([\![\phi_1]\!]_>, [\![\phi_2]\!]_>)$

$u \vdash \phi[n] \iff [\![\phi]\!]_>(u) \geq n$

$$[\![\phi[n]]\!]_>(u) = \begin{cases} +\infty & \text{if } [\![\phi]\!]_>(u) \geq n \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\phi \wedge \phi[n]]\!]_>(u) = \begin{cases} [\![\phi]\!]_>(u) & \text{if } [\![\phi]\!]_>(u) \geq n \\ 0 & \text{otherwise} \end{cases}$$

$\phi[n]$ refines ϕ by forbidding words of value $< n$

If $n \leq \sup[\![\phi]\!]_>$ then $\sup[\![\phi \wedge \phi[n]]\!]_> = \sup[\![\phi]\!]_>$

Refinement loop for upper bound

Repeat until $\mathcal{A}_\phi$ has no more accepting runs

Find an acc. run ρ in $\mathcal{A}_\phi$
Let $n = \mathrm{Val}_>(\rho)$
Then $n \leq \sup[\![\mathcal{A}]\!]_>$
Update $\phi \leftarrow \phi \wedge \phi[n]$ and restart

Refinement loop for upper bound

Repeat until $\mathcal{A}_\phi$ has no more accepting runs

Find an acc. run ρ in $\mathcal{A}_\phi$
Let $n = \mathrm{Val}_>(\rho)$
Then $n \leq \sup[\![\mathcal{A}]\!]_>$
Update $\phi \leftarrow \phi \wedge \phi[n]$ and restart

Requires $\sup[\![A_{\phi_0}]\!]_> < \infty$

$\sup[\![A_{\phi_0}]\!]_> = \sup_{\rho \text{ acc. run}} \mathrm{Val}_>(\rho)$
Guarantees that every found ρ has a finite value

Refinement loop for upper bound

Repeat until $\mathcal{A}_\phi$ has no more accepting runs

Find an acc. run ρ in $\mathcal{A}_\phi$
Let $n = \mathrm{Val}_>(\rho)$
Then $n \leq \sup[\![\mathcal{A}]\!]_>$
Update $\phi \leftarrow \phi \wedge \phi[n]$ and restart

Requires $\sup[\![A_{\phi_0}]\!]_> < \infty$

$\sup[\![A_{\phi_0}]\!]_> = \sup_{\rho \text{ acc. run}} \mathrm{Val}_>(\rho)$
Guarantees that every found ρ has a finite value

- easy to implement: formulae manipulations + Büchi EC
- refinement: less and less behaviors

Loop for boundedness + exact value

```
INPUT: φ_0 and M
```

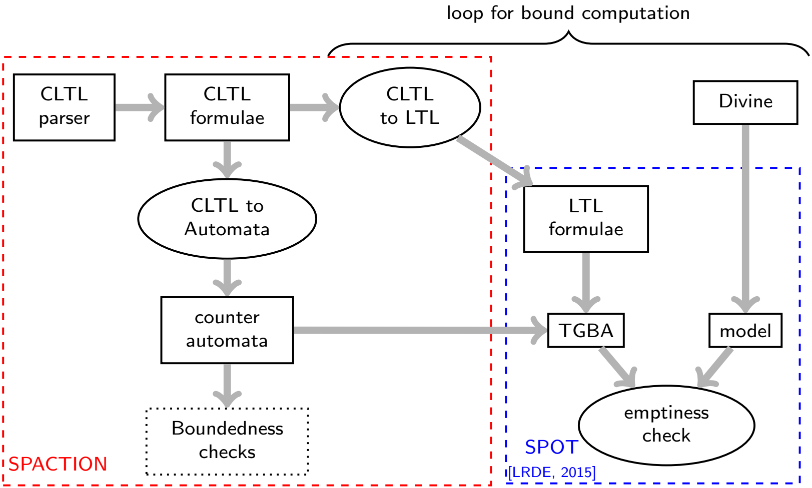
INPUT: ϕ_0 **and** $\mathcal{M}$

$B \leftarrow |\mathcal{A}_{\phi_0}| \times |\mathcal{M}|$
// *if finite*, $\sup[\![\mathcal{A}_{\phi_0}]\!]_> \leq |\mathcal{A}_{\phi_0} \times \mathcal{M}| \leq B$
$\phi \leftarrow \phi_0$
$n \leftarrow 0$

while $(\mathcal{L}(\mathcal{A}_\phi \times \mathcal{M}) \neq \emptyset)$ {
  $\rho \leftarrow$ an accepting run in $\mathcal{A}_\phi \times \mathcal{M}$
  $n \leftarrow \mathrm{Val}_>(\rho)$

  if $(n > B)$
    return unbounded
  else
    $\phi \leftarrow \phi_0 \wedge \phi_0[n]$
}
return n

Our tool Spaction



Conclusion

CLTL = nice extension of LTL

- expressivity
- counter automata = nice extension of ω-automata
- separate the logics from the model
- algorithm for bound computation
  - relies on ω-automata emptiness check
- working tool (in progress)

What's next?

- examples and use cases
- more abstractions and refinements
  - boundedness: smaller automata, fewer counters
  - exact value: smaller synchronized products
- relaxed versions: $\sup[\![\mathcal{A}_1]\!]_> \leq n * \sup[\![\mathcal{A}_2]\!]_>$
- variants and generalization

Bibliography

Bojańczyk, M. and Colcombet, T. (2006).
Bounds in ω-regularity.
In *Logic in Computer Science, 2006 21st Annual IEEE Symposium on*, pages 285–296. IEEE.

Colcombet, T. (2009).
The theory of stabilisation monoids and regular cost functions.
In *Automata, languages and programming*, pages 139–150. Springer.

Kuperberg, D. (2012).
Étude de classes de fonctions de coût régulières.
PhD thesis, Université Paris Diderot.

Kuperberg, D. and Boom, M. V. (2012).
On the expressive power of cost logics over infinite words.
In *Automata, Languages, and Programming*, pages 287–298. Springer.

LRDE (2005-2015).
SPOT home page.
`http://spot.lip6.fr/`.