Averaging in LTL

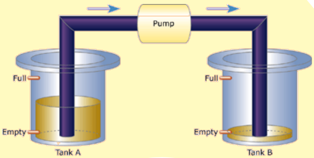
Patricia Bouyer Nicolas Markey Raj Mohan Matteplackel

LSV, CNRS & ENS Cachan, France

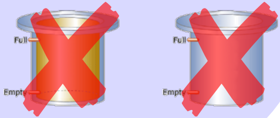
Outline

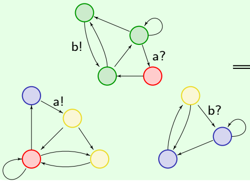
Model-checking

system:



property:





model-checking algorithm

$\mathbf{AG}(\neg\texttt{B.overfull} \wedge \neg\texttt{B.dried\_up})$

yes/no

Boolean verification

Standard model-checking

Strict dichotomy between correct and incorrect systems

Boolean verification

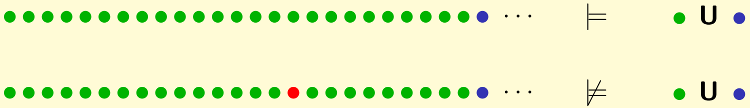
Standard model-checking

Strict dichotomy between correct and incorrect systems

Boolean verification

Standard model-checking

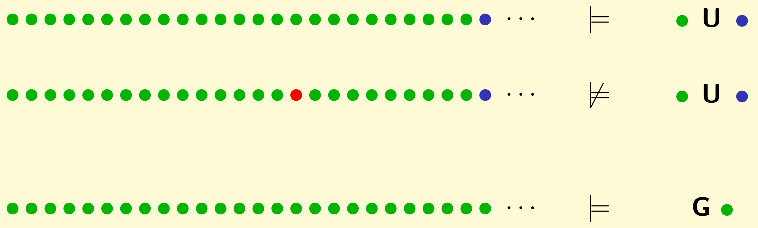
Strict dichotomy between correct and incorrect systems



Boolean verification

Standard model-checking

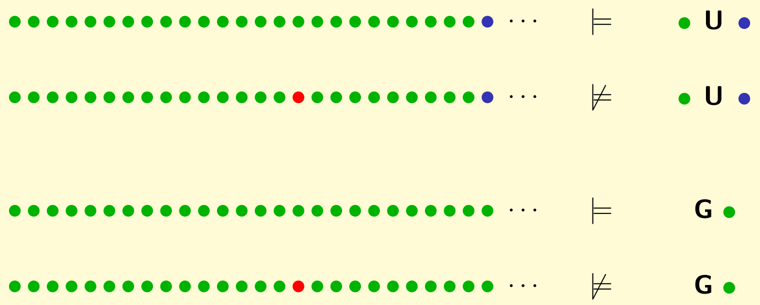
Strict dichotomy between correct and incorrect systems



Boolean verification

Standard model-checking

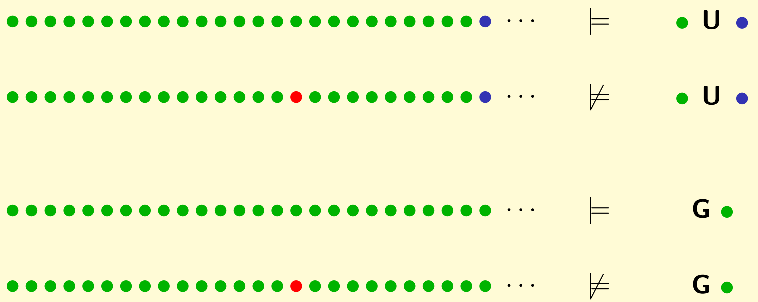
Strict dichotomy between correct and incorrect systems



Boolean verification

Standard model-checking

Strict dichotomy between correct and incorrect systems



$\rightsquigarrow$ this Boolean approach might be too crude

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

☞ give a value in $[0, 1]$ instead of a Boolean value!

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

☞ give a value in $[0, 1]$ instead of a Boolean value!

high value to $\bullet$ **U** $\bullet$

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

☞ give a value in $[0, 1]$ instead of a Boolean value!

 high value to • **U** •

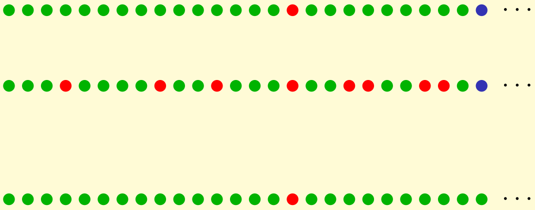
 smaller value to • **U** •

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

☞ give a value in $[0, 1]$ instead of a Boolean value!



high value to • **U** •

smaller value to • **U** •

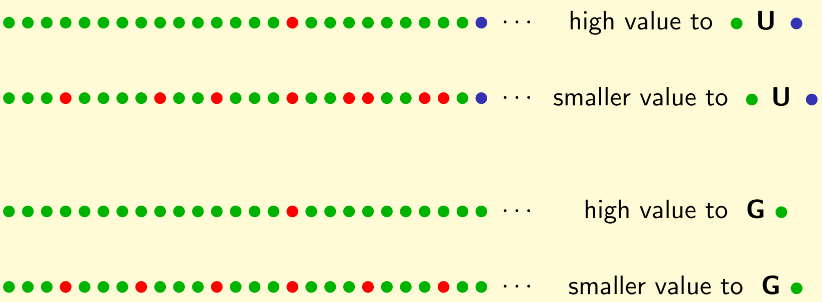
high value to **G** •

Towards quantitative verification

Quantitative model-checking

Measure the accuracy of a system w.r.t. a property.

☞ give a value in $[0,1]$ instead of a Boolean value!



Related work

Quantitative verification

- value of executions given by weighted automata [DCH10,...]
- accuracy of a model given by a distance to another model or to a specification (e.g. simulation distance [CHR12], model measuring [HO13])
- quantitative specification languages/logics: standard in probabilistic model-checking (e.g. CSL and PCTL logics)
- ...

Related work

Quantitative verification

- value of executions given by weighted automata [DCH10,...]
- accuracy of a model given by a distance to another model or to a specification (e.g. simulation distance [CHR12], model measuring [HO13])
- quantitative specification languages/logics: standard in probabilistic model-checking (e.g. CSL and PCTL logics)
- ...

Related work – cont'd

Quantitative logics based on LTL

- Logics yielding finitely many values
  - min/max extension of LTL to quant. Kripke structures [FLS08]
  - extension of LTL with functions [ABK13]
  - $\rightsquigarrow$ the model-checking is decidable

Related work – cont'd

Quantitative logics based on LTL

- Logics yielding finitely many values
  - min/max extension of LTL to quant. Kripke structures [FLS08]
  - extension of LTL with functions [ABK13]

  $\rightsquigarrow$ the model-checking is decidable
- LTL over weighted Kripke structures using numerical assertions over weight variables [BCHK11,THHY12,BGM14]

  $\rightsquigarrow$ various decidability/undecidability results

Related work – cont'd

Quantitative logics based on LTL

- Logics yielding finitely many values
  - min/max extension of LTL to quant. Kripke structures [FLS08]
  - extension of LTL with functions [ABK13]

  $\rightsquigarrow$ the model-checking is decidable
- LTL over weighted Kripke structures using numerical assertions over weight variables [BCHK11,THHY12,BGM14]

  $\rightsquigarrow$ various decidability/undecidability results
- Frequency LTL [BDL12]
  - boolean interpretation with quantitative constraints
  - $\pi \models \bullet\, \mathbf{U}^{1/2} \bullet$ if at least half of the positions satisfy $\bullet$ before $\bullet$

  $\rightsquigarrow$ the model-checking is undecidable

Related work – cont'd

Quantitative logics based on LTL

- Logics yielding finitely many values
  - min/max extension of LTL to quant. Kripke structures [FLS08]
  - extension of LTL with functions [ABK13]

  $\rightsquigarrow$ the model-checking is decidable
- LTL over weighted Kripke structures using numerical assertions over weight variables [BCHK11,THHY12,BGM14]

  $\rightsquigarrow$ various decidability/undecidability results
- Frequency LTL [BDL12]
  - boolean interpretation with quantitative constraints
  - $\pi \models \bullet\, \mathbf{U}^{1/2} \bullet$ if at least half of the positions satisfy $\bullet$ before $\bullet$

  $\rightsquigarrow$ the model-checking is undecidable
- Discounted LTL [ABK14]
  - the until modality discounts over the future

  $\rightsquigarrow$ the (threshold) model-checking is decidable, undecidable with an extra local average operator

Outline

1. Introduction
2. Average-LTL
3. Results
4. Conclusion

The logic Average-LTL

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathrm{U}}\,\varphi \mid \widetilde{\mathrm{G}}\,\varphi.$$

The logic Average-LTL

$$\varphi \ ::= \ p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathbf{U}}\,\varphi \mid \widetilde{\mathbf{G}}\,\varphi.$$

$$\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!])
\end{aligned}$$

The logic Average-LTL

$$\varphi \ ::= \ p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi \,\mathbf{U}\, \varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathbf{U}}\,\varphi \mid \widetilde{\mathbf{G}}\,\varphi.$$

$$\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \mathbf{X}\,\varphi]\!] &= [\![\pi_{\geq 1}, \varphi]\!] \\
[\![\pi, \mathbf{G}\,\varphi]\!] &= \inf_i\big([\![\pi_{\geq i}, \varphi]\!]\big) \\
[\![\pi, \psi\,\mathbf{U}\,\varphi]\!] &= \sup_i\Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0\leq j<i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)
\end{aligned}$$

More

The logic Average-LTL

$$\varphi \ ::= \ p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi \ \mathbf{U}\ \varphi \mid \mathbf{G}\,\varphi \mid \varphi\, \widetilde{\mathrm{U}}\, \varphi \mid \widetilde{\mathrm{G}}\,\varphi.$$

$$\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \mathbf{X}\,\varphi]\!] &= [\![\pi_{\geq 1}, \varphi]\!] \\
[\![\pi, \mathbf{G}\,\varphi]\!] &= \inf_i \big([\![\pi_{\geq i}, \varphi]\!]\big) \\
[\![\pi, \psi\ \mathbf{U}\ \varphi]\!] &= \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big) \\
[\![\pi, \widetilde{\mathrm{G}}\,\varphi]\!] &= \liminf_{i \to \infty} \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \varphi]\!]\big)/i
\end{aligned}$$

The logic Average-LTL

$$\varphi \;::=\; p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathbf{U}}\,\varphi \mid \widetilde{\mathbf{G}}\,\varphi.$$

$$\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi,\psi_1]\!], [\![\pi,\psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi,\psi_1]\!], [\![\pi,\psi_2]\!]) \\
[\![\pi, \mathbf{X}\,\varphi]\!] &= [\![\pi_{\geq 1}, \varphi]\!] \\
[\![\pi, \mathbf{G}\,\varphi]\!] &= \inf_i \left([\![\pi_{\geq i}, \varphi]\!]\right) \\
[\![\pi, \psi\,\mathbf{U}\,\varphi]\!] &= \sup_i \Big(\min\left([\![\pi_{\geq i}, \varphi]\!], \min_{0\leq j<i}([\![\pi_{\geq j}, \psi]\!])\right)\Big) \\
[\![\pi, \widetilde{\mathbf{G}}\,\varphi]\!] &= \liminf_{i\to\infty}\left(\textstyle\sum_{j=0}^{j<i}[\![\pi_{\geq j}, \varphi]\!]\right)/i \\
[\![\pi, \psi\,\widetilde{\mathbf{U}}\,\varphi]\!] &= \max\Big([\![\pi,\varphi]\!], \sup_{i>0}\left(\min\left([\![\pi_{\geq i},\varphi]\!], \left(\textstyle\sum_{j=0}^{j<i}[\![\pi_{\geq j},\psi]\!]\right)/i\right)\right)\Big)
\end{aligned}$$

More

The logic Average-LTL

$$\varphi \ ::= \ p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\ \mathbf{U}\ \varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathbf{U}}\,\varphi \mid \widetilde{\mathbf{G}}\,\varphi.$$

$$\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \mathbf{X}\,\varphi]\!] &= [\![\pi_{\geq 1}, \varphi]\!] \\
[\![\pi, \mathbf{G}\,\varphi]\!] &= \inf_i\left([\![\pi_{\geq i}, \varphi]\!]\right) \\
[\![\pi, \psi\ \mathbf{U}\ \varphi]\!] &= \sup_i \Big(\min\left([\![\pi_{\geq i}, \varphi]\!], \min_{0\leq j<i}([\![\pi_{\geq j}, \psi]\!])\right)\Big) \\
[\![\pi, \widetilde{\mathbf{G}}\,\varphi]\!] &= \liminf_{i\to\infty}\left(\textstyle\sum_{j=0}^{j<i}[\![\pi_{\geq j}, \varphi]\!]\right)/i \\
[\![\pi, \psi\,\widetilde{\mathbf{U}}\,\varphi]\!] &= \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\left(\min\left([\![\pi_{\geq i}, \varphi]\!], \left(\textstyle\sum_{j=0}^{j<i}[\![\pi_{\geq j}, \psi]\!]\right)/i\right)\right)\Big)
\end{aligned}$$

For a Kripke structure $\mathcal{K}$,

$$[\![\mathcal{K}, \varphi]\!] = \sup_{\pi \text{ path in } \mathcal{K}} [\![\pi, \varphi]\!]$$

The logic Average-LTL

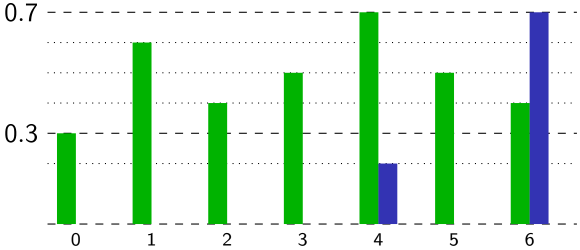
$$\varphi \ ::= \ p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\ \mathbf{U}\ \varphi \mid \mathbf{G}\,\varphi \mid \varphi\,\widetilde{\mathbf{U}}\,\varphi \mid \widetilde{\mathbf{G}}\,\varphi.$$

$$
\begin{aligned}
[\![\pi, \psi_1 \vee \psi_2]\!] &= \max([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \psi_1 \wedge \psi_2]\!] &= \min([\![\pi, \psi_1]\!], [\![\pi, \psi_2]\!]) \\
[\![\pi, \mathbf{X}\,\varphi]\!] &= [\![\pi_{\geq 1}, \varphi]\!] \\
[\![\pi, \mathbf{G}\,\varphi]\!] &= \inf_i \left([\![\pi_{\geq i}, \varphi]\!]\right) \\
[\![\pi, \psi\ \mathbf{U}\ \varphi]\!] &= \sup_i \Big(\min\left([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\right)\Big) \\
[\![\pi, \widetilde{\mathbf{G}}\,\varphi]\!] &= \liminf_{i \to \infty} \left(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \varphi]\!]\right)/i \\
[\![\pi, \psi\,\widetilde{\mathbf{U}}\,\varphi]\!] &= \max \Big([\![\pi, \varphi]\!], \sup_{i>0}\left(\min\left([\![\pi_{\geq i}, \varphi]\!], \left(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\right)/i\right)\right)\Big)
\end{aligned}
$$

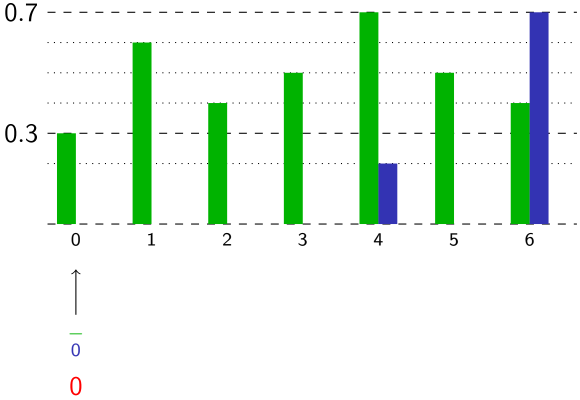
For a Kripke structure $\mathcal{K}$,

$$[\![\mathcal{K}, \varphi]\!] = \sup_{\pi \text{ path in } \mathcal{K}} [\![\pi, \varphi]\!]$$

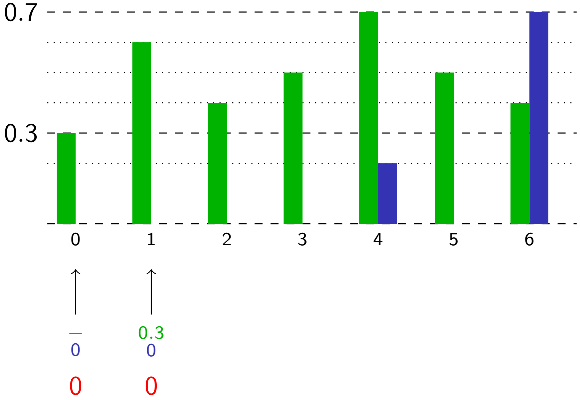
Continue



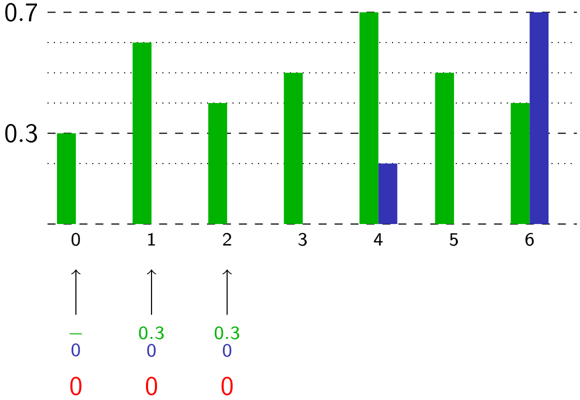
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)$$



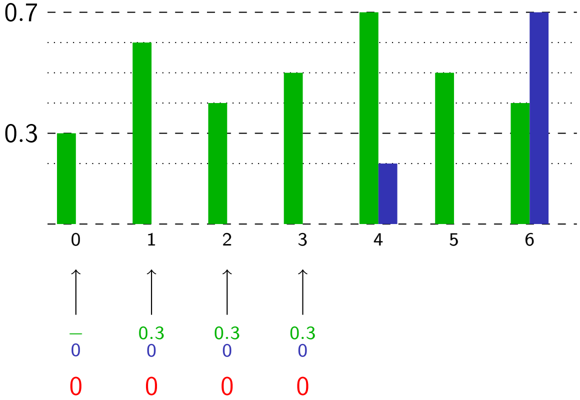
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \left(\min\left([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\right)\right)$$



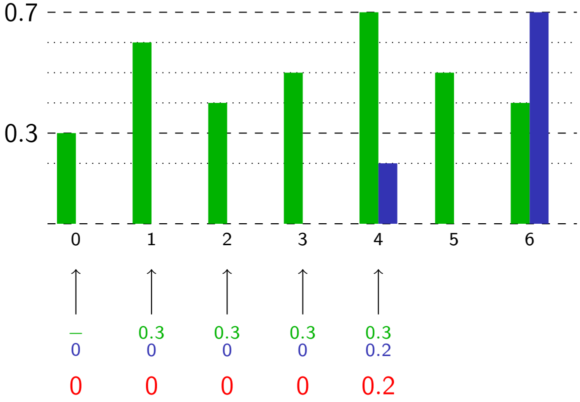
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0\leq j<i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)$$



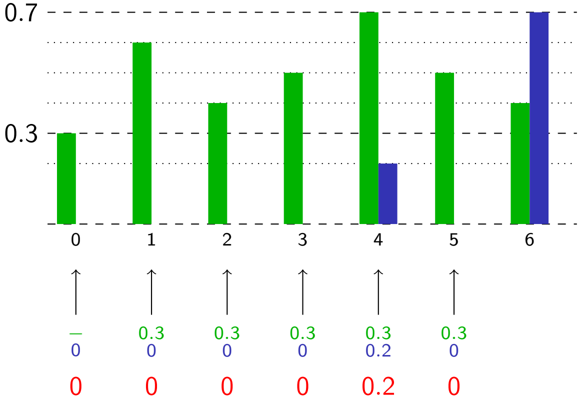
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \mathsf{sup}_i \left(\mathsf{min}\big([\![\pi_{\geq i}, \varphi]\!], \mathsf{min}_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\right)$$



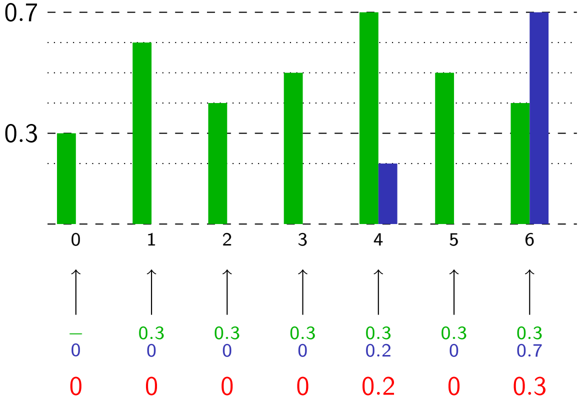
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)$$



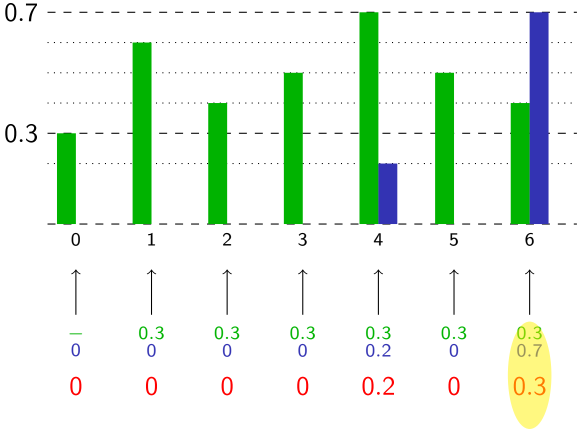
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)$$



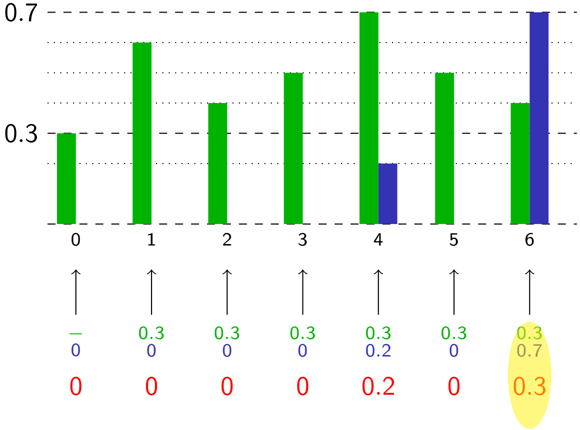
$$\llbracket \pi, \psi \ \mathbf{U} \ \varphi \rrbracket = \sup_i \Big(\min\big(\llbracket \pi_{\geq i}, \varphi \rrbracket, \min_{0 \leq j < i}(\llbracket \pi_{\geq j}, \psi \rrbracket)\big)\Big)$$



$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big)$$

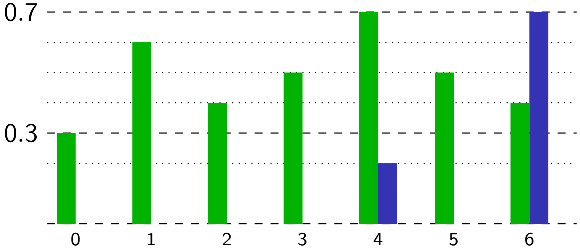


$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big) = 0.3$$

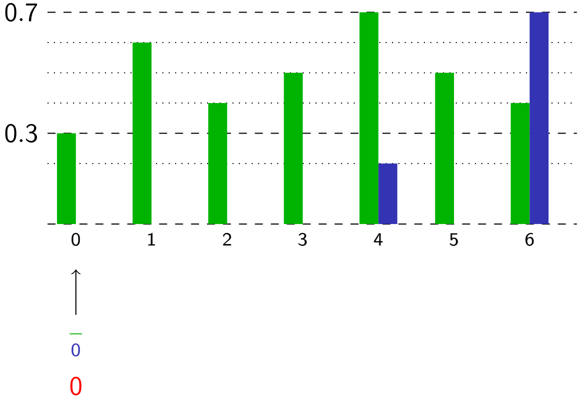


Back

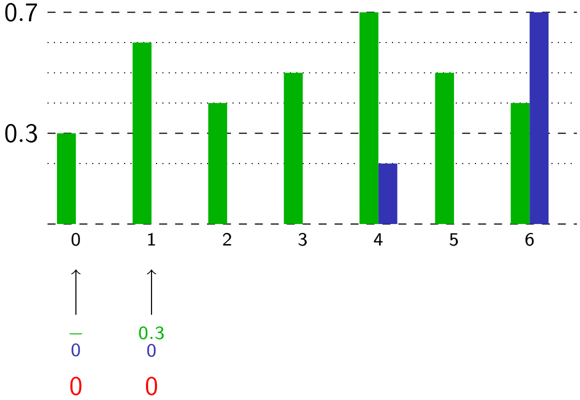
$$[\![\pi, \psi \ \mathbf{U} \ \varphi]\!] = \sup_i \Big(\min\big([\![\pi_{\geq i}, \varphi]\!], \min_{0 \leq j < i}([\![\pi_{\geq j}, \psi]\!])\big)\Big) = 0.3$$



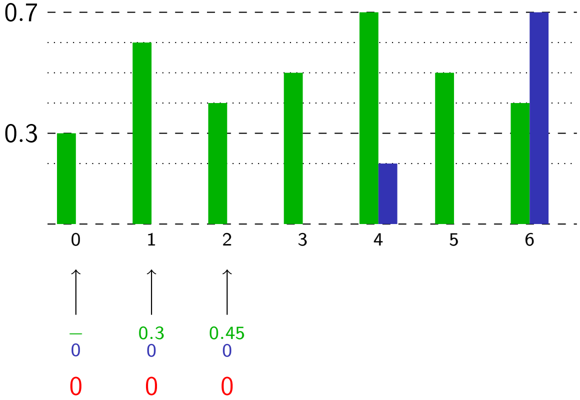
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\big)\big)\Big)$$



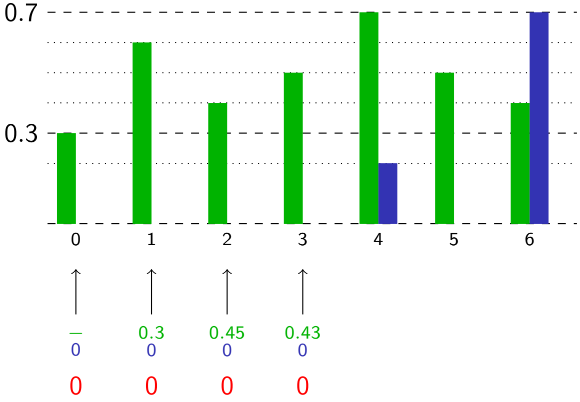
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\left([\![\pi, \varphi]\!], \sup_{i>0}\Big(\min\Big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\Big)\Big)\right)$$



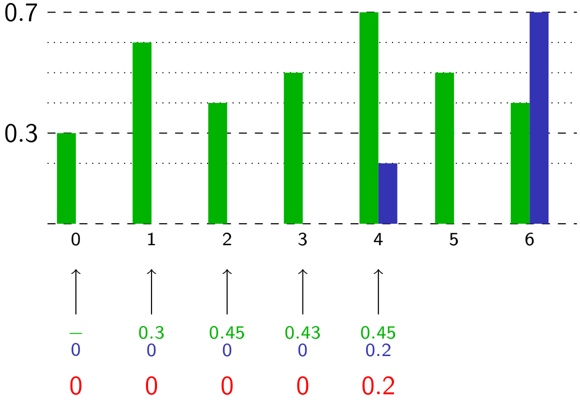
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\big)\big)\Big)$$



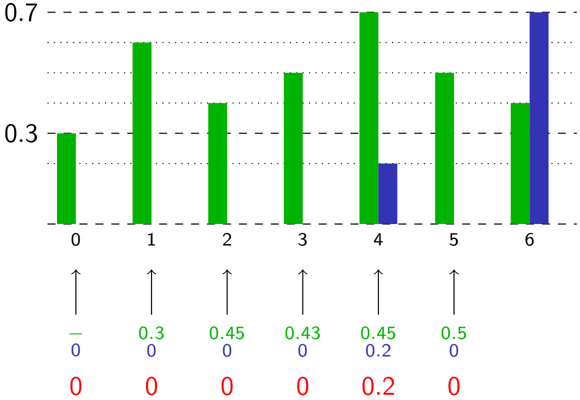
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi,\varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i},\varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j},\psi]\!]\big)/i\big)\big)\Big)$$



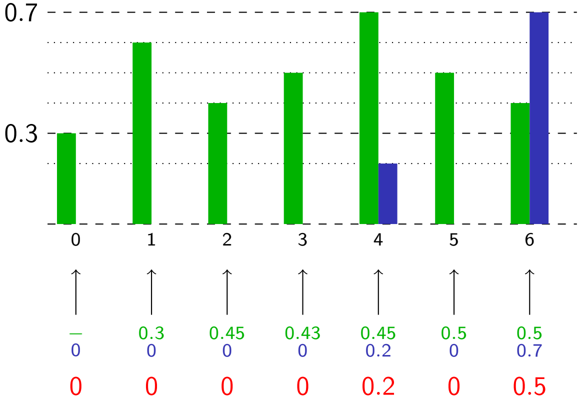
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\left([\![\pi, \varphi]\!], \sup_{i>0}\left(\min\left([\![\pi_{\geq i}, \varphi]\!], \left(\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\right)/i\right)\right)\right)$$



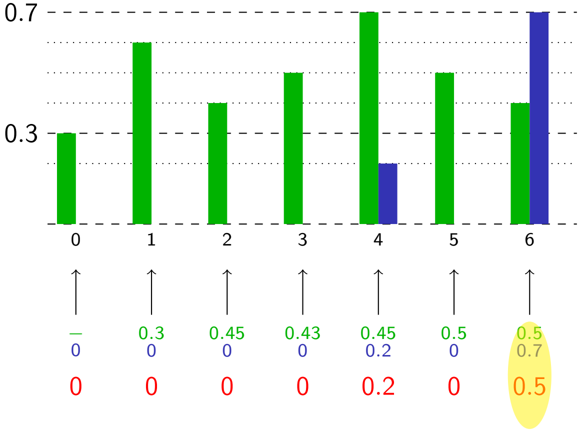
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\big)\big)\Big)$$



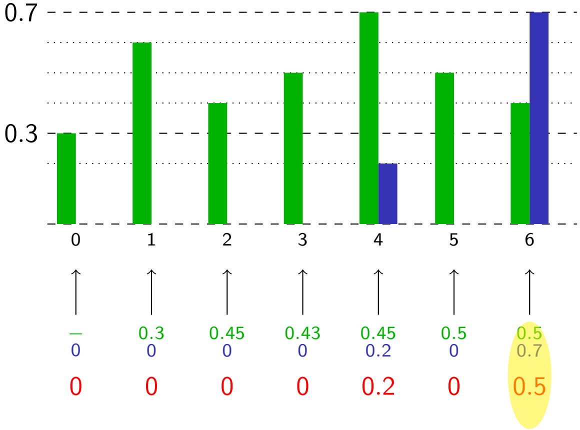
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\big)\big)\Big)$$



$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\left([\![\pi,\varphi]\!], \mathsf{sup}_{i>0}\left(\min\left([\![\pi_{\geq i},\varphi]\!], \left(\textstyle\sum_{j=0}^{j<i}[\![\pi_{\geq j},\psi]\!]\right)/i\right)\right)\right)$$



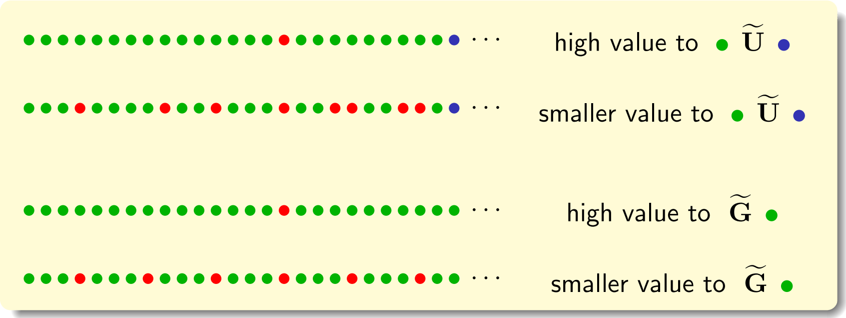
$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\Big([\![\pi, \varphi]\!], \sup_{i>0}\big(\min\big([\![\pi_{\geq i}, \varphi]\!], \big(\textstyle\sum_{j=0}^{j<i} [\![\pi_{\geq j}, \psi]\!]\big)/i\big)\big)\Big) = 0.5$$



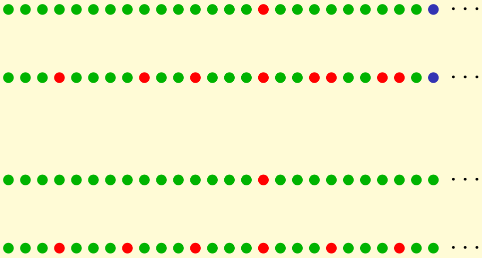
Back

$$[\![\pi, \psi \,\widetilde{\mathbf{U}}\, \varphi]\!] = \max\left([\![\pi,\varphi]\!], \sup_{i>0}\left(\min\left([\![\pi_{\geq i},\varphi]\!], \left(\sum_{j=0}^{j<i} [\![\pi_{\geq j},\psi]\!]\right)/i\right)\right)\right) = 0.5$$

Back to the introductory examples



Back to the introductory examples



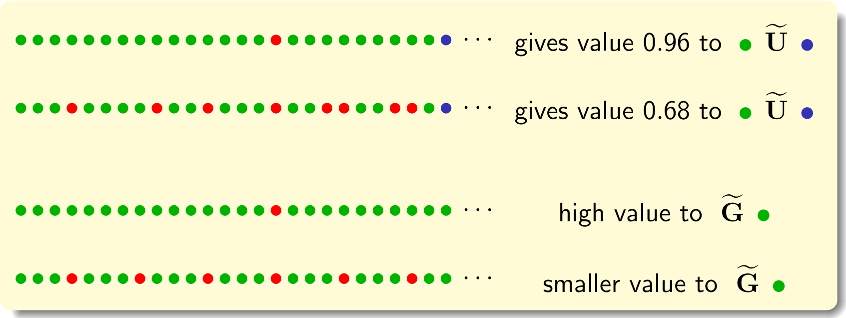
gives value 0.96 to ● $\widetilde{\mathbf{U}}$ ●

smaller value to ● $\widetilde{\mathbf{U}}$ ●

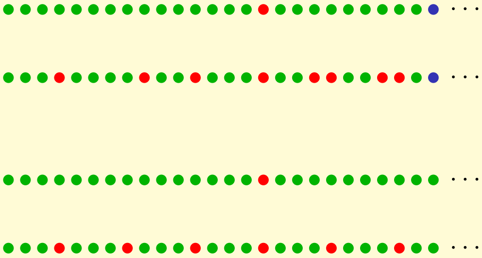
high value to $\widetilde{\mathbf{G}}$ ●

smaller value to $\widetilde{\mathbf{G}}$ ●

Back to the introductory examples



Back to the introductory examples



gives value 0.96 to $\bullet\ \widetilde{\mathbf{U}}\ \bullet$

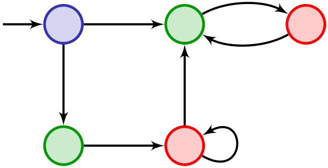
gives value 0.68 to $\bullet\ \widetilde{\mathbf{U}}\ \bullet$

gives value 1 to $\widetilde{\mathbf{G}}\ \bullet$

smaller value to $\widetilde{\mathbf{G}}\ \bullet$

Back to the introductory examples

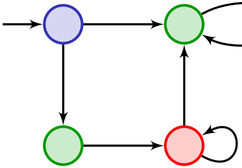
Examples of computations in Kripke structure $\mathcal{K}$



$$[\![\mathcal{K}, {\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = ?$$

- Formula ${\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}$:
  - $[\![{\color{blue}\bullet} \cdot {\color{green}\bullet} \cdot {\color{red}\bullet}^{\omega}, {\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = 1$

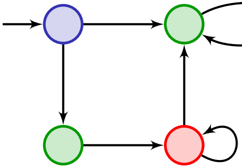
Examples of computations in Kripke structure $\mathcal{K}$



$[\![\mathcal{K}, \bullet \, \widetilde{\mathbf{U}} \, \bullet]\!] = 1$

- Formula $\bullet \, \widetilde{\mathbf{U}} \, \bullet$:
  - $[\![\bullet \cdot \bullet \cdot \bullet^{\omega}, \bullet \, \widetilde{\mathbf{U}} \, \bullet]\!] = 1$

Examples of computations in Kripke structure $\mathcal{K}$

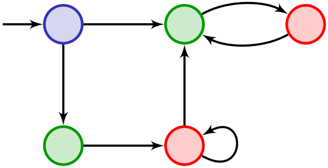


$[\![\mathcal{K}, \color{blue}{\bullet}\, \widetilde{\mathbf{U}}\, \color{green}{\bullet}]\!] = 1$

$[\![\mathcal{K}, \color{red}{\bullet}\, \widetilde{\mathbf{U}}\, \color{green}{\bullet}]\!] = ?$

- Formula $\color{blue}{\bullet}\, \widetilde{\mathbf{U}}\, \color{green}{\bullet}$:
  - $[\![\color{blue}{\bullet} \cdot \color{green}{\bullet} \cdot \color{red}{\bullet}^{\omega}, \color{blue}{\bullet}\, \widetilde{\mathbf{U}}\, \color{green}{\bullet}]\!] = 1$
- Formula $\color{red}{\bullet}\, \widetilde{\mathbf{U}}\, \color{green}{\bullet}$:

Examples of computations in Kripke structure $\mathcal{K}$

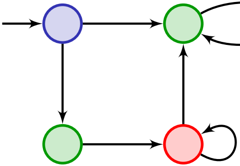


$[\![\mathcal{K}, \textcolor{blue}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}]\!] = 1$

$[\![\mathcal{K}, \textcolor{red}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}]\!] = ?$

- Formula $\textcolor{blue}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{blue}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}]\!] = 1$
- Formula $\textcolor{red}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{red}{\bullet}\ \widetilde{\mathbf{U}}\ \textcolor{green}{\bullet}]\!] = 0$

Examples of computations in Kripke structure $\mathcal{K}$

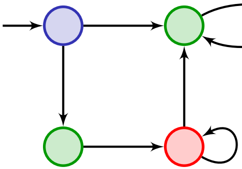


$[\![\mathcal{K}, \textcolor{blue}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}]\!] = 1$

$[\![\mathcal{K}, \textcolor{red}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}]\!] = ?$

- Formula $\textcolor{blue}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{blue}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}]\!] = 1$
- Formula $\textcolor{red}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{red}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}]\!] = 0$
  - $[\![\textcolor{blue}{\bullet} \cdot (\textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet})^{\omega}, \textcolor{red}{\bullet} \, \widetilde{\mathbf{U}} \, \textcolor{green}{\bullet}]\!] = \sup \{n/(2n+1) \mid n \in \mathbb{N}^*\} = 1/2.$
    Note that the value $1/2$ is not reached by any prefix.

Examples of computations in Kripke structure $\mathcal{K}$

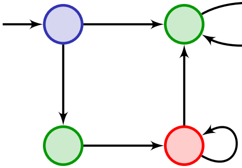


$[\![\mathcal{K}, {\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = 1$

$[\![\mathcal{K}, {\color{red}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = ?$

- Formula ${\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}$:
  - $[\![{\color{blue}\bullet} \cdot {\color{green}\bullet} \cdot {\color{red}\bullet}^{\omega}, {\color{blue}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = 1$
- Formula ${\color{red}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}$:
  - $[\![{\color{blue}\bullet} \cdot {\color{green}\bullet} \cdot {\color{red}\bullet}^{\omega}, {\color{red}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = 0$
  - $[\![{\color{blue}\bullet} \cdot ({\color{green}\bullet} \cdot {\color{red}\bullet})^{\omega}, {\color{red}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = \sup\{n/(2n+1) \mid n \in \mathbb{N}^*\} = 1/2$.
    Note that the value $1/2$ is not reached by any prefix.
  - Write $\pi'_k = {\color{blue}\bullet} \cdot {\color{green}\bullet} \cdot {\color{red}\bullet}^{k} \cdot ({\color{green}\bullet} \cdot {\color{red}\bullet})^{\omega}$. Then:
    $[\![\pi'_k, {\color{red}\bullet}\,\widetilde{\mathbf{U}}\,{\color{green}\bullet}]\!] = \sup\{(k+n)/(k+2n+2) \mid n \in \mathbb{N}\} = k/(k+2)$

Examples of computations in Kripke structure $\mathcal{K}$



$$[\![\mathcal{K}, \textcolor{blue}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = 1$$

$$[\![\mathcal{K}, \textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = 1$$

- Formula $\textcolor{blue}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{blue}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = 1$
- Formula $\textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}$:
  - $[\![\textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{\omega}, \textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = 0$
  - $[\![\textcolor{blue}{\bullet} \cdot (\textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet})^{\omega}, \textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = \sup\{n/(2n+1) \mid n \in \mathbb{N}^*\} = 1/2.$
    Note that the value $1/2$ is not reached by any prefix.
  - Write $\pi'_k = \textcolor{blue}{\bullet} \cdot \textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet}^{k} \cdot (\textcolor{green}{\bullet} \cdot \textcolor{red}{\bullet})^{\omega}$. Then:
    $[\![\pi'_k, \textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = \sup\{(k+n)/(k+2n+2) \mid n \in \mathbb{N}\} = k/(k+2)$

Hence $[\![\mathcal{K}, \textcolor{red}{\bullet}\,\widetilde{\mathbf{U}}\,\textcolor{green}{\bullet}]\!] = 1$, even though no run witnesses that value.

Outline

1. Introduction
2. Average-LTL
3. Results
4. Conclusion

Our results

For any threshold $\bowtie c$:

- Existence problem: given $\mathcal{K}$, φ, is there π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] \bowtie c$?
- Value problem: given $\mathcal{K}$, φ, does $[\![\mathcal{K}, \varphi]\!] \bowtie c$?

Our results

For any threshold $\bowtie c$:

- Existence problem: given $\mathcal{K}$, φ, is there π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] \bowtie c$?
- Value problem: given $\mathcal{K}$, φ, does $[\![\mathcal{K}, \varphi]\!] \bowtie c$?

Theorem

The existence and value problems are undecidable (for every threshold of the form $\bowtie 1/2$).

Our results

For any threshold $\bowtie c$:

- Existence problem: given $\mathcal{K}$, φ, is there π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] \bowtie c$?
- Value problem: given $\mathcal{K}$, φ, does $[\![\mathcal{K}, \varphi]\!] \bowtie c$?

Theorem

The existence and value problems are undecidable (for every threshold of the form $\bowtie 1/2$).

Not always a link between the existence and the value problems:

- $[\![\mathcal{K}, \varphi]\!] > 1/2$ iff there is π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] > 1/2$

Our results

For any threshold $\bowtie c$:

- Existence problem: given $\mathcal{K}$, φ, is there π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] \bowtie c$?
- Value problem: given $\mathcal{K}$, φ, does $[\![\mathcal{K}, \varphi]\!] \bowtie c$?

Theorem

The existence and value problems are undecidable (for every threshold of the form $\bowtie 1/2$).

Not always a link between the existence and the value problems:

- $[\![\mathcal{K}, \varphi]\!] > 1/2$ iff there is π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] > 1/2$
- The same equivalence does not hold for other thresholds $\bowtie 1/2$

Our results

For any threshold $\bowtie c$:

- Existence problem: given $\mathcal{K}$, φ, is there π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] \bowtie c$?
- Value problem: given $\mathcal{K}$, φ, does $[\![\mathcal{K}, \varphi]\!] \bowtie c$?

Theorem

The existence and value problems are undecidable (for every threshold of the form $\bowtie 1/2$).

Not always a link between the existence and the value problems:

- $[\![\mathcal{K}, \varphi]\!] > 1/2$ iff there is π in $\mathcal{K}$ s.t. $[\![\pi, \varphi]\!] > 1/2$
- The same equivalence does not hold for other thresholds $\bowtie 1/2$
  - $[\![\mathcal{K}, \varphi]\!] = 1/2$ iff there exists a sequence $(\pi^n)_{n\in\mathbb{N}}$ s.t. $[\![\pi^n, \varphi]\!] \leq 1/2$ for every n, and $\lim_{n\to\infty}[\![\pi^n, \varphi]\!] = 1/2$

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$
- Execution encoded as a run (concatenation of the above)

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$
- Execution encoded as a run (concatenation of the above)
- Structure of $\mathcal{M}$ and zero tests encoded in a Kripke structure $\mathcal{K}_{\mathcal{M}}$

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$
- Execution encoded as a run (concatenation of the above)
- Structure of $\mathcal{M}$ and zero tests encoded in a Kripke structure $\mathcal{K}_{\mathcal{M}}$
- Checking counter values requires Average-LTL

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$
- Execution encoded as a run (concatenation of the above)
- Structure of $\mathcal{M}$ and zero tests encoded in a Kripke structure $\mathcal{K}_{\mathcal{M}}$
- Checking counter values requires Average-LTL

  $(q, n_0, n_1) \to (q', n'_0, n'_1)$ keeps counter values unchanged whenever for every $\alpha \in \{1 + n_0 + n_1, 1 + n_0 + n'_1, 1 + n'_0 + n_1, 1 + n'_0 + n'_1\}$, it holds that:

$$\frac{\alpha}{(1 + n_0 + n_1) + (1 + n'_0 + n'_1)} = \frac{1}{2}$$

Undecidability of the existence problem

Encoding of a two-counter machine $\mathcal{M}$

- Configuration (q, n_0, n_1) encoded in unary: $q \bullet^{n_0} \bullet^{n_1}$
- Execution encoded as a run (concatenation of the above)
- Structure of $\mathcal{M}$ and zero tests encoded in a Kripke structure $\mathcal{K}_{\mathcal{M}}$
- Checking counter values requires Average-LTL

  $(q, n_0, n_1) \rightarrow (q', n_0', n_1')$ keeps counter values unchanged whenever for every $\alpha \in \{1 + n_0 + n_1, 1 + n_0 + n_1', 1 + n_0' + n_1, 1 + n_0' + n_1'\}$, it holds that:

$$\frac{\alpha}{(1 + n_0 + n_1) + (1 + n_0' + n_1')} = \frac{1}{2}$$

  I.e.

$$[\![q \bullet^{n_0} \bullet^{n_1} q' \bullet^{n_0'} \bullet^{n_1'} q'', \psi_\alpha \,\widetilde{\mathbf{U}}\, Q]\!] = \frac{1}{2}$$

  where $\psi_\alpha = q' \vee \bullet \vee \bullet$ when $\alpha = 1 + n_0 + n_1'$

Undecidability of the existence problem (cont'd)

Global reduction

- Formula to be checked:

$$\mathtt{halt}_{\mathcal{M}} = \mathbf{F}\, q_{halt} \wedge \mathbf{G}\, \mathtt{consec}_{\mathcal{M}}$$

where (roughly) $\mathtt{consec}_{\mathcal{M}} = \bigwedge_{q,\alpha} \left(q \to \psi_\alpha \,\widetilde{\mathbf{U}}\, Q\right)$

Undecidability of the existence problem (cont'd)

Global reduction

- Formula to be checked:

$$\mathtt{halt}_{\mathcal{M}} = \mathbf{F}\, q_{halt} \wedge \mathbf{G}\, \mathtt{consec}_{\mathcal{M}}$$

 where (roughly) $\mathtt{consec}_{\mathcal{M}} = \bigwedge_{q,\alpha} \left(q \rightarrow \psi_\alpha \, \widetilde{\mathbf{U}} \, Q \right)$

- The two counter machine halts iff there is an execution π such that $[\![\pi, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$

Undecidability of the existence problem (cont'd)

Global reduction

- Formula to be checked:

$$\mathtt{halt}_{\mathcal{M}} = \mathbf{F}\, q_{halt} \wedge \mathbf{G}\, \mathtt{consec}_{\mathcal{M}}$$

  where (roughly) $\mathtt{consec}_{\mathcal{M}} = \bigwedge_{q,\alpha} \left(q \to \psi_\alpha \,\widetilde{\mathbf{U}}\, Q \right)$
- The two counter machine halts iff there is an execution π such that $[\![\pi, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$

$\rightsquigarrow$ This yields the undecidability of the existence problem for threshold $= 1/2$

What about the value problem?

- If $\mathcal{M}$ halts, then $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$

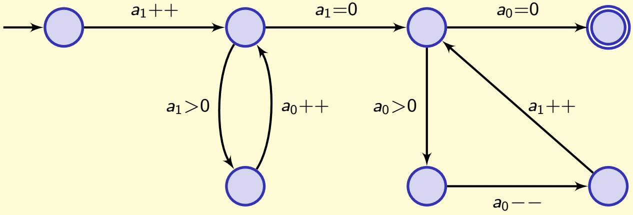
What about the value problem?

- If $\mathcal{M}$ halts, then $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$
- It might be the case that $\mathcal{M}$ does not halt but $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$

What about the value problem?

- If $\mathcal{M}$ halts, then $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$
- It might be the case that $\mathcal{M}$ does not halt but $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$

$\mathcal{M}$ does not halt but $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{halt}_{\mathcal{M}}]\!] = 1/2$



☞ We need to be careful with machines having such a converging phenomenon

Arguments for the undecidability proof

Technical lemmas

- For a finite invalid run, if the first invalid consecution assigns a value smaller than $1/2 - 1/n$ to $\texttt{consec}_{\mathcal{M}}$, then the value of formula $\mathbf{G}\,\texttt{consec}_{\mathcal{M}}$ along the run is smaller than $1/2 - 1/n$.

Arguments for the undecidability proof

Technical lemmas

- For a finite invalid run, if the first invalid consecution assigns a value smaller than $1/2 - 1/n$ to $\texttt{consec}_{\mathcal{M}}$, then the value of formula $\mathbf{G}\,\texttt{consec}_{\mathcal{M}}$ along the run is smaller than $1/2 - 1/n$.

  We assume two halting states: accept and reject.

Arguments for the undecidability proof

Technical lemmas

- For a finite invalid run, if the first invalid consecution assigns a value smaller than $1/2 - 1/n$ to $\mathtt{consec}_{\mathcal{M}}$, then the value of formula $\mathbf{G}\,\mathtt{consec}_{\mathcal{M}}$ along the run is smaller than $1/2 - 1/n$.

  We assume two halting states: accept and reject.

- If $[\![\mathcal{K}_{\mathcal{M}}, \mathtt{accept}_{\mathcal{M}}]\!] = 1/2$ but no run gives value $1/2$ to formula $\mathtt{accept}_{\mathcal{M}}$, then the unique valid run is infinite.

The diagonal argument

B det. Turing machine can either accept, reject, or not halt

$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

The diagonal argument

B det. Turing machine can either accept, reject, or not halt

$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \texttt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The diagonal argument

B det. Turing machine can either accept, reject, or not halt

$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

The diagonal argument

$$\begin{array}{ll} & B \text{ det. Turing machine can either accept, reject, or not halt} \\ \rightarrow & \mathcal{M}(B) \text{ two-counter machine which simulates } B \text{ on } B \end{array}$$

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \texttt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$,

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

The diagonal argument

$$\begin{array}{ll} & B \text{ det. Turing machine can either accept, reject, or not halt} \\ \rightarrow & \mathcal{M}(B) \text{ two-counter machine which simulates } B \text{ on } B \end{array}$$

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$$\begin{array}{ll} \mathcal{C}(B): & \text{Simulate } \mathcal{T}_{\mathcal{H}} \text{ on } B; \\ & \text{If } \mathcal{T}_{\mathcal{H}} \text{ accepts } B \text{ then reject, otherwise accept} \end{array}$$

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$:

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$,

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$.

The diagonal argument

$$\begin{array}{ll} & B \text{ det. Turing machine can either accept, reject, or not halt} \\ \rightarrow & \mathcal{M}(B) \text{ two-counter machine which simulates } B \text{ on } B \end{array}$$

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$$\begin{array}{ll} \mathcal{C}(B): & \text{Simulate } \mathcal{T}_{\mathcal{H}} \text{ on } B; \\ & \text{If } \mathcal{T}_{\mathcal{H}} \text{ accepts } B \text{ then reject, otherwise accept} \end{array}$$

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \mathit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept,

The diagonal argument

B det. Turing machine can either accept, reject, or not halt

$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$,

The diagonal argument

$$\begin{array}{ll} & B \text{ det. Turing machine can either accept, reject, or not halt} \\ \rightarrow & \mathcal{M}(B) \text{ two-counter machine which simulates } B \text{ on } B \end{array}$$

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$$\begin{array}{ll} \mathcal{C}(B): & \text{Simulate } \mathcal{T}_{\mathcal{H}} \text{ on } B; \\ & \text{If } \mathcal{T}_{\mathcal{H}} \text{ accepts } B \text{ then reject, otherwise accept} \end{array}$$

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.
- $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] = 1/2$.

The diagonal argument

$$\begin{array}{ll} & B \text{ det. Turing machine can either accept, reject, or not halt} \\ \rightarrow & \mathcal{M}(B) \text{ two-counter machine which simulates } B \text{ on } B \end{array}$$

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \mathit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \mathit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \mathit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.
- $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] = 1/2$. However since $\mathcal{C}$ does not accept $\mathcal{C}$, the unique valid computation of $\mathcal{M}(\mathcal{C})$ is either infinite or rejecting.

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.
- $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] = 1/2$. However since $\mathcal{C}$ does not accept $\mathcal{C}$, the unique valid computation of $\mathcal{M}(\mathcal{C})$ is either infinite or rejecting. Applying the lemma on previous slide, it is infinite,

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \mathtt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.
- $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \mathtt{accept}_{\mathcal{M}(\mathcal{C})}]\!] = 1/2$. However since $\mathcal{C}$ does not accept $\mathcal{C}$, the unique valid computation of $\mathcal{M}(\mathcal{C})$ is either infinite or rejecting. Applying the lemma on previous slide, it is infinite, which contradicts the fact that $\mathcal{C}$ always terminates.

The diagonal argument

B det. Turing machine can either accept, reject, or not halt
$\rightarrow$ $\mathcal{M}(B)$ two-counter machine which simulates B on B

We define the program

$$\mathcal{H} : B \mapsto \begin{cases} \textit{accept} & \text{if } [\![\mathcal{K}_{\mathcal{M}(B)}, \texttt{accept}_{\mathcal{M}(B)}]\!] = 1/2 \\ \textit{reject} & \text{otherwise} \end{cases}$$

The function $\mathcal{H}$ is not computable.

Towards a contradiction, assume it is computable by det. Turing machine $\mathcal{T}_{\mathcal{H}}$, and define the program:

$\mathcal{C}(B)$: Simulate $\mathcal{T}_{\mathcal{H}}$ on B;
If $\mathcal{T}_{\mathcal{H}}$ accepts B then reject, otherwise accept

Program $\mathcal{C}$ is deterministic hence we can run $\mathcal{C}$ on $\mathcal{C}$
Program $\mathcal{C}$ always terminates:

- Assume $\mathcal{C}$ accepts $\mathcal{C}$: this means that $\mathcal{H}(\mathcal{C}) = \textit{reject}$, hence $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \texttt{accept}_{\mathcal{M}(\mathcal{C})}]\!] < 1/2$. This implies $\mathcal{M}(\mathcal{C})$ does not accept, and therefore $\mathcal{C}$ does not accept $\mathcal{C}$, contradiction: $\mathcal{C}$ rejects $\mathcal{C}$.
- $[\![\mathcal{K}_{\mathcal{M}(\mathcal{C})}, \texttt{accept}_{\mathcal{M}(\mathcal{C})}]\!] = 1/2$. However since $\mathcal{C}$ does not accept $\mathcal{C}$, the unique valid computation of $\mathcal{M}(\mathcal{C})$ is either infinite or rejecting. Applying the lemma on previous slide, it is infinite, which contradicts the fact that $\mathcal{C}$ always terminates.

Therefore, $\mathcal{H}$ is not computable.

Outline

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
    - No nesting of quantitative modalities in the proof is required
    - Decidable twist: only using discounting

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]
- Undecidability also holds for several approximation problems

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]
- Undecidability also holds for several approximation problems

- However, original and interesting (we believe) diagonal argument for proving the undecidability

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]
- Undecidability also holds for several approximation problems

- However, original and interesting (we believe) diagonal argument for proving the undecidability
  - We did not manage to simulate the halting problem for Turing machines directly. What does that mean for the undecidability?

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]
- Undecidability also holds for several approximation problems

- However, original and interesting (we believe) diagonal argument for proving the undecidability
  - We did not manage to simulate the halting problem for Turing machines directly. What does that mean for the undecidability?
  - Proof technics re-used in the context of timed systems

Conclusion and current work

- We have proposed a natural and expressive quant. extension of LTL
- Unfortunately the model-checking is undecidable
  - No nesting of quantitative modalities in the proof is required
  - Decidable twist: only using discounting
  - Note: LTL with mean-payoff constraints is decidable [THHY12]
- Undecidability also holds for several approximation problems

- However, original and interesting (we believe) diagonal argument for proving the undecidability
  - We did not manage to simulate the halting problem for Turing machines directly. What does that mean for the undecidability?
  - Proof technics re-used in the context of timed systems
  - Might be useful in other quantitative contexts?