Computing Branching Distances Using Quantitative Games

Uli Fahrenberg Axel Legay Karin Quaas

École polytechnique, Palaiseau, France

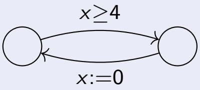
Université Catholique de Louvain, Belgium

Universitä Leipzig, Germany

ICTAC 2019

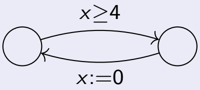
Quantitative Analysis

Quantitative *Models*



Quantitative Quantitative Analysis

Quantitative *Models*

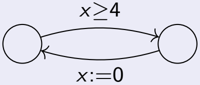


Quantitative *Logics*

$$\mathrm{Pr}_{\leq .1}(\Diamond \mathit{error})$$

Quantitative Quantitative Quantitative Analysis

Quantitative *Models*



Quantitative *Logics*

$$\Pr_{\leq .1}(\Diamond \mathit{error})$$

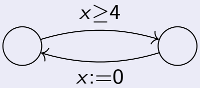
Quantitative *Verification*

$$[\![\phi]\!](s) = 3.14$$

$$d(s, t) = 42$$

Quantitative Quantitative Quantitative Analysis

Quantitative *Models*



Quantitative *Logics*

$$\mathrm{Pr}_{\leq .1}(\Diamond \mathit{error})$$

Quantitative *Verification*

$$[\![\phi]\!](s) = 3.14$$
$$d(s, t) = 42$$

Boolean world	**"Quantification"**
Trace equivalence $\equiv$	Linear distances d_L
Bisimilarity $\sim$	Branching distances d_B
$s \sim t$ implies $s \equiv t$	$d_L(s, t) \leq d_B(s, t)$
$s \models \phi$ or $s \not\models \phi$	$[\![\phi]\!](s)$ is a quantity
$s \sim t$ iff $\forall \phi : s \models \phi \Leftrightarrow t \models \phi$	$d_B(s, t) = \sup_\phi d\big([\![\phi]\!](s), [\![\phi]\!](t)\big)$

Quantitative Quantitative Quantitative Analysis

Problem: For processes with quantities, lots of different ways to measure distance

- point-wise $D(\sigma, \tau) = \sup_i |\sigma_i - \tau_i|$
- accumulating $D(\sigma, \tau) = \sum_i |\sigma_i - \tau_i|$
- limit-average $D(\sigma, \tau) = \limsup_N \frac{1}{N} \sum_{i=0}^N |\sigma_i - \tau_i|$
- discounted $D(\sigma, \tau) = \sum_i \lambda^i |\sigma_i - \tau_i|$
- maximum-lead $D(\sigma, \tau) = \sup_N |\sum_{i=0}^N \sigma_i - \sum_{i=0}^N \tau_i|$
- Cantor $D(\sigma, \tau) = 1/(1 + \inf\{j \mid \sigma_j \neq \tau_j\})$
- discrete $D(\sigma, \tau) = 0$ if $\sigma = \tau$; ∞ otherwise
- etc.

Upshot

Three ideas:

- For an application, it is easiest to define distance between system traces (executions)
- Use games to convert these *linear* distances to *branching* distances
- (this paper) Use other games to compute branching distances

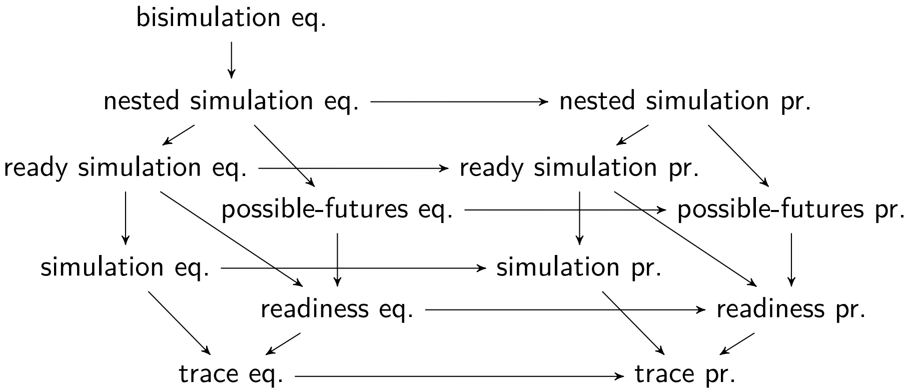
The Linear-Time–Branching-Time Spectrum

van Glabbeek, 2001 (excerpt):

$$
\begin{tikzcd}
 & \text{bisimulation eq.} \arrow[d] & \\
 & \text{nested simulation eq.} \arrow[dl] \arrow[dr] & \\
\text{ready simulation eq.} \arrow[d] \arrow[drr] & & \text{possible-futures eq.} \arrow[d] \\
\text{simulation eq.} \arrow[dr] & & \text{readiness eq.} \arrow[dl] \\
 & \text{trace eq.} &
\end{tikzcd}
$$

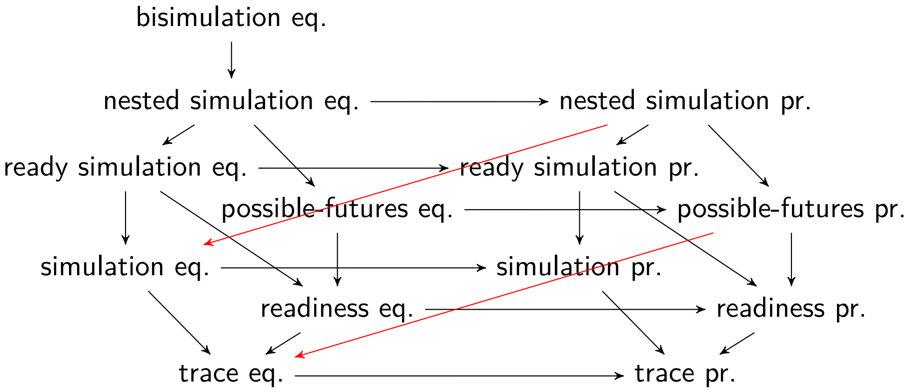
The Linear-Time–Branching-Time Spectrum

van Glabbeek, 2001 (excerpt):

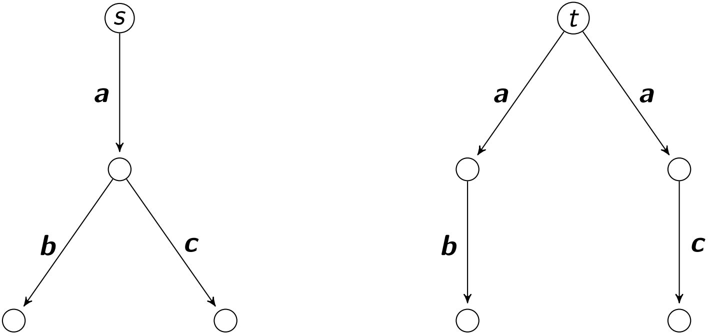


The Linear-Time–Branching-Time Spectrum

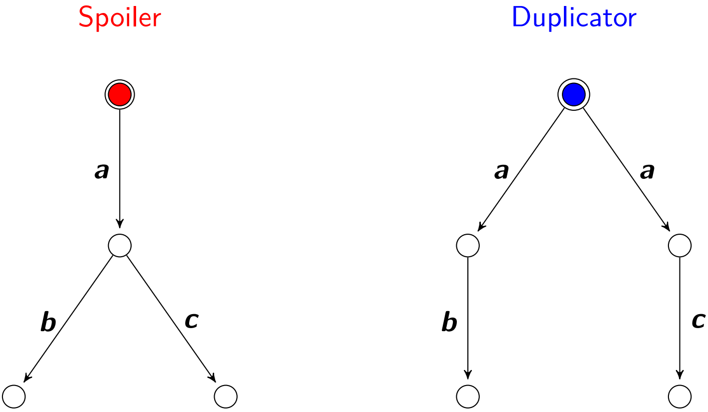
van Glabbeek, 2001 (excerpt):



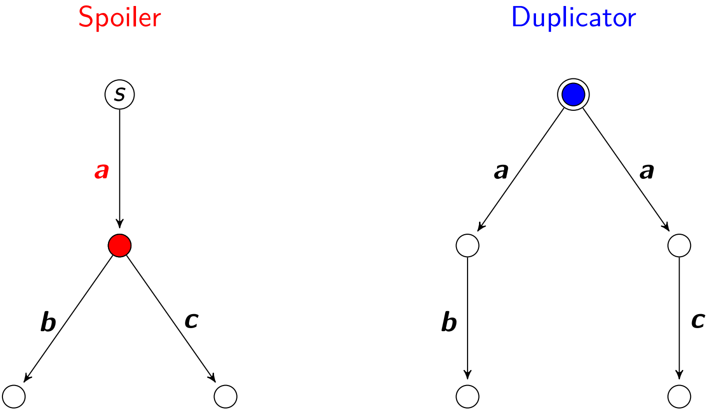
The Simulation Game



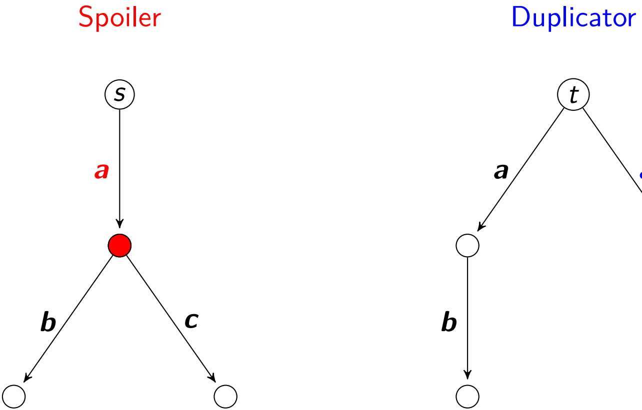
The Simulation Game



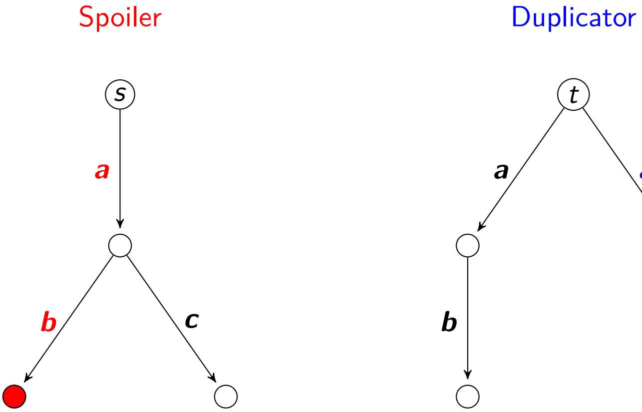
The Simulation Game



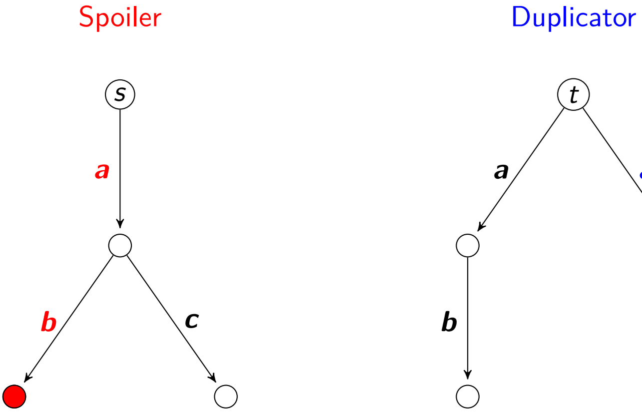
The Simulation Game



The Simulation Game



The Simulation Game



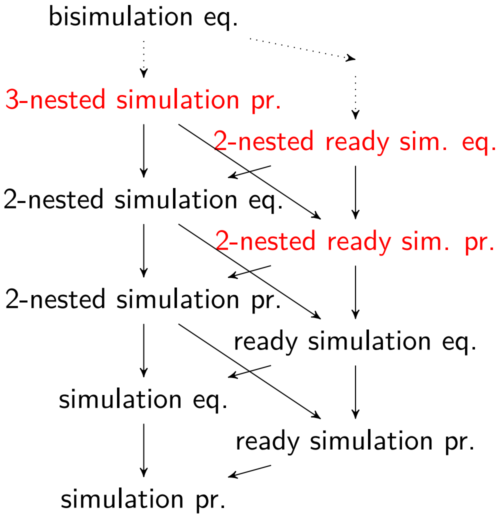
Spoiler wins

The Simulation Game

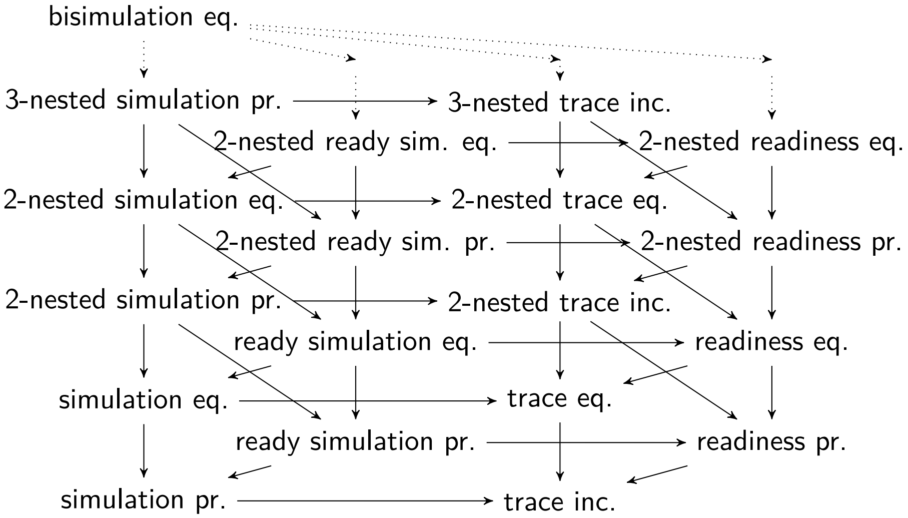
1. Player 1 ("Spoiler") chooses edge from s (leading to s')
2. Player 2 ("Duplicator") chooses matching edge from t (leading to t')
3. Game continues from configuration s', t'

ω. If Player 2 can always answer: YES, t simulates s.
Otherwise: NO

The Linear-Time–Branching-Time Spectrum, Reordered



The Linear-Time–Branching-Time Spectrum, Reordered



The Simulation Game, Revisited

1. Player 1 chooses edge from s (leading to s')
2. Player 2 chooses matching edge from t (leading to t')
3. Game continues from configuration s', t'

ω. If Player 2 can always answer: YES, t simulates s.
Otherwise: NO

Or, as an Ehrenfeucht-Fraïssé game:

1. Player 1 chooses edge from s (leading to s')
2. Player 2 chooses edge from t (leading to t')
3. Game continues from new configuration s', t'

ω. At the end (maybe after infinitely many rounds!),
compare the chosen traces:
If the trace chosen by t matches the one chosen by s: YES
Otherwise: NO

Quantitative Ehrenfeucht-Fraïssé Games

The quantitative setting:

- Assume we have a way, possibly application-determined, to measure distances of (finite or infinite) traces
- a hemimetric $D : (\sigma, \tau) \mapsto D(\sigma, \tau) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$

The quantitative Ehrenfeucht-Fraïssé game:

1. Player 1 chooses edge from s (leading to s')
2. Player 2 chooses edge from t (leading to t')
3. Game continues from new configuration s', t'

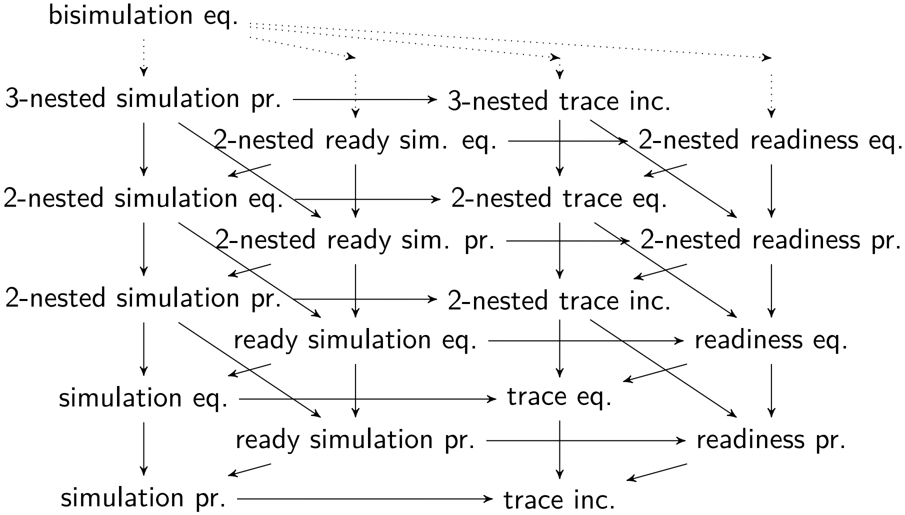
ω. At the end, compare the chosen traces σ, τ:
The simulation distance from s to t is defined to be $D(\sigma, \tau)$

- Player 1 plays to maximize $D(\sigma, \tau)$; Player 2 plays to minimize $D(\sigma, \tau)$

This can be done for all the games in the LTBT spectrum.

The Quantitative Linear-Time–Branching-Time Spectrum

For any trace distance $D : (\sigma, \tau) \mapsto D(\sigma, \tau) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$:



Transfer Theorem

- Given two equivalences or preorders in the *qualitative* setting for which there is a *counter-example* which separates them, then the two corresponding distances are topologically inequivalent
- (under certain mild conditions for the trace distance)
- (The proof uses precisely the same counter-example)

Path-Building Games

- Have seen how branching distances can be defined using a type of "double path-building game"
- But now, how to compute them?
- Nothing in the literature about computing values of double path-building games. . .
- On the other hand, people know how to compute values of (single) path-building games!
  - reachability games; discounted games; mean-payoff games, . . .
- So, let's convert our double path-building games to single path-building games

From Double to Single Path-Building Games

- Let $D : (\sigma, \tau) \mapsto D(\sigma, \tau) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ be a hemimetric on traces
- Assume that we have functions val_D and f_D such that always,

$$D(\sigma, \tau) = \mathsf{val}_D(0, f_D(\sigma_0, \tau_0), 0, f_D(\sigma_1, \tau_1), 0, \dots)$$

- Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS
- Construct a game $\mathcal{U} = \mathcal{U}(\mathcal{S}, \mathcal{S}') = (U_1 \cup U_2, u_0, \rightarrowtriangle)$ by

$$U_1 = S \times S' \qquad U_2 = S \times S' \times \Sigma \qquad u_0 = (i, i')$$

$$\begin{aligned} \rightarrowtriangle = &\{(s, s') \xrightarrow{0} (t, s', a) \mid (s, a, t) \in T\} \\ &\cup \{(t, s', a) \xrightarrow{f_D(a, a')} (t, t') \mid (s', a', t') \in T'\} \end{aligned}$$

- Path-building game: players alternate to build path π
- Player 1 plays to maximize $\mathsf{val}_D(\pi)$; Player 2 plays to minimize $\mathsf{val}_D(\pi)$

Computing Distances Using Path-Building Games

$\mathcal{U}(\mathcal{S}, \mathcal{S}') = (U_1 \cup U_2, u_0, \rightharpoonup)$:

$$U_1 = S \times S' \qquad U_2 = S \times S' \times \Sigma \qquad u_0 = (i, i')$$

$$\begin{aligned} \rightharpoonup = &\{(s, s') \xrightarrow{0} (t, s', a) \mid (s, a, t) \in T\} \\ &\cup \{(t, s', a) \xrightarrow{f_D(a,a')} (t, t') \mid (s', a', t') \in T'\} \end{aligned}$$

Theorem

The value of $\mathcal{U}(\mathcal{S}, \mathcal{S}')$ is the simulation distance from $\mathcal{S}$ to $\mathcal{S}'$.

Computing Distances Using Path-Building Games, contd.

$\mathcal{V}(\mathcal{S}, \mathcal{S}') = (V_1 \cup V_2, v_0, \twoheadrightarrow)$:

$$V_1 = S \times S' \qquad V_2 = S \times S' \times \Sigma \times \{1, 2\} \qquad v_0 = (i, i')$$

$$\begin{aligned}
\twoheadrightarrow = {} & \{(s, s') \xrightarrow{0} (t, s', a, 1) \mid (s, a, t) \in T\} \\
& \cup \{(s, s') \xrightarrow{0} (s, t', a', 2) \mid (s', a', t') \in T'\} \\
& \cup \{(t, s', a, 1) \xrightarrow{f_D(a,a')} (t, t') \mid (s', a', t') \in T'\} \\
& \cup \{(s, t', a', 2) \xrightarrow{f_D(a,a')} (t, t') \mid (s, a, t) \in T\}
\end{aligned}$$

Theorem

The value of $\mathcal{V}(\mathcal{S}, \mathcal{S}')$ is the bisimulation distance between $\mathcal{S}$ and $\mathcal{S}'$.

- Similar constructions for all distances in the linear-time–branching-time spectrum

Coda: Computing the Values of Path-Building Games

$\mathcal{U}(\mathcal{S}, \mathcal{S}') = (U_1 \cup U_2, u_0, \rightharpoonup)$:

$$U_1 = S \times S' \qquad U_2 = S \times S' \times \Sigma \qquad u_0 = (i, i')$$

$$\begin{aligned} \rightharpoonup = \{(s, s') \xrightarrow{0} (t, s', a) \mid (s, a, t) \in T\} \\ \cup \{(t, s', a) \xrightarrow{f_D(a,a')} (t, t') \mid (s', a', t') \in T'\} \end{aligned}$$

- discrete distance: reachability game
- point-wise distance: weighted reachability game
- discounted distance: discounted game
- limit-average distance: mean-payoff game
- maximum-lead distance: energy game
- Cantor distance: iterated reachability game

Conclusion & Further Work

- A general method to define linear and branching system distances using double path-building games
- A general method to compute linear and branching system distances using (single) path-building games

- Application to real-time and hybrid systems
- Quantitative specification theories
- Quantitative LTBT with silent moves?
- What about probabilistic systems?

Quantitative EF Games: The Gory Details – 1

- Configuration of the game: (π, ρ): π the Player-1 choices up to now; ρ the Player-2 choices
- Strategy: mapping from configurations to next moves
  - Θ_i: set of Player-i strategies
- Simulation strategy: Player-1 moves allowed from end of π
- Bisimulation strategy: Player-1 moves allowed from end of π or end of ρ
  - (hence π and ρ are generally not paths – "mingled paths")
- Pair of strategies $\implies$ (possibly infinite) sequence of configurations
- Take the limit; unmingle $\implies$ pair of (possibly infinite) traces (σ, τ)
- Bisimulation distance: $\sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} D(\sigma, \tau)$
- Simulation distance: $\sup_{\theta_1 \in \Theta_1^0} \inf_{\theta_2 \in \Theta_2} D(\sigma, \tau)$ (restricting Player 1's capabilities)

Quantitative EF Games: The Gory Details – 2

- Blind Player-1 strategies: depend only on the end of ρ
  - ("cannot see Player-2 moves")
  - $\tilde{\Theta}_1$: set of blind Player-1 strategies
- Trace inclusion distance: $\sup_{\theta_1 \in \tilde{\Theta}_1^0} \inf_{\theta_2 \in \Theta_2} D(\sigma, \tau)$
- For nesting: count the number of times Player 1 choses edge from end of ρ
  - Θ_1^k: k choices from end of ρ allowed
- Nested simulation distance: $\sup_{\theta_1 \in \Theta_1^1} \inf_{\theta_2 \in \Theta_2} D(\sigma, \tau)$
- Nested trace inclusion distance: $\sup_{\theta_1 \in \tilde{\Theta}_1^1} \inf_{\theta_2 \in \Theta_2} D(\sigma, \tau)$
- For ready: allow extra "I'll see you" Player-1 transition from end of ρ