Language inclusion for Büchi automata

The problem

Given two non-deterministic Büchi automata (NBA) $A, B$, does it hold that

$$L(A) \subseteq L(B)?$$

L.I. in model-checking

- LTL model-checking
- regular model-checking
- size-change termination analysis
- ...

A hard problem

- PSPACE-complete
- already for trace inclusion (for two LTS)
- sometimes even undecidable (e.g. TA)
Recent advances on language inclusion

- antichain algorithms [Doyen et al]
- bisimulation up to techniques [Bonchi and Pous]
- Ramsey or antichain algorithms enhanced with simulation preorder [Holik et al]
- automata minimization
  - using SAT solving [Ehlers]
  - quotienting, pruning [Clemente and Mayr]
- http://www.languageinclusion.ord

Our goal: design better simulations ⇒ get better algorithms
Simulations step by step

Why simulation matters

- under-approximates language inclusion
- can be used for automata minimization
- computing simulation is typically fast
  in time $O(|A| \cdot |B|)$ for two NFA $A, B$

Looking for better simulations

- delayed simulation [Etessami, Wilke, and Schuller]
  $\Rightarrow$ NBA quotient is language equivalent
- multi-pebble simulation [Etessami]
  $\Rightarrow$ better approximates language inclusion
- multi-letter simulation
  [Clemente and Mayr] [Hutagalung et al]
  $\Rightarrow$ sometimes faster than multi-pebble
Fair simulation

two players, infinite game

- Spoiler moves a pebble in $A$, Duplicator in $B$

- in every round:
  1. Spoiler chooses a letter $a$
  2. Spoiler moves his pebble along an $a$ transition
  3. Duplicator moves her pebble along an $a$ transition

- if a player gets stuck, he/she loses
- otherwise, two infinite runs

Fairness condition: Duplicator wins if
  - either her run is accepting,
  - or Spoiler’s one is not.
two players, infinite game

- Spoiler moves a pebble in $A$, Duplicator in $B$
- in every round:
  1. Spoiler chooses $k$ letters $a_1, \ldots, a_k$
  2. Spoiler moves his pebble along transitions $a_1, \ldots a_k$
  3. Duplicator moves her pebble along transitions $a_1, \ldots a_k$

- if a player gets stuck, he/she looses
- otherwise, two infinite runs
- Fairness condition: Duplicator wins if
  - either her run is accepting,
  - or Spoiler’s one is not.
Example

- \((A) \subseteq L(B)\) but \(A\) is not simulated by \(B\)
- \(A\) is 2-letters simulated by \(B\)
$k$-letters simulations in practice

- cheap to compute for small $k$
- on most benchmarks, significantly better than 1-letter simulation (up to 100 times faster than Rabit v1.0)

- untractable for large $k$
- Duplicator looses in “easy cases”
Example

- $L(A) \subseteq L(B)$
- $A$ is not $k$-letters simulated by $B$, for any $k$. 
Introducing continuous simulation

players now share a FIFO buffer
in every round:

1. Spoiler chooses $a \in \Sigma$, and adds it to the buffer
2. Spoiler moves his pebble along an $a$ transition
3. Duplicator can decide to
   - either skip her turn
   - or pop $b$ from the buffer, and move along a $b$ transition
Introducing continuous simulation

players now share a FIFO buffer in every round:

1. Spoiler chooses $a \in \Sigma$, and adds it to the buffer
2. Spoiler moves his pebble along an $a$ transition
3. Duplicator can decide to
   - either skip her turn
   - or pop $b$ from the buffer, and move along a $b$ transition

Winning condition

- if a player gets stuck, he/she looses
- if Spoiler makes no accepting run, he looses
otherwise,
- if Duplicator eventually only skips her turn, she looses
- otherwise, apply fairness condition on the two infinite runs
Example

Winning strategy for Duplicator:
- skip her first turn
- at round $i$, play Spoiler’s letter in round $i - 1$
Fairness and buffer boundedness

- Spoiler must eventually leave the $a$ loop
- Duplicator wins
- Buffer can grow arbitrarily large
Continuous simulation and continuity

- \( \text{ARuns}(A) \): the set of accepting runs of \( A \)
- \( \text{ARuns}(A) \) is a metric space
- \( f : \text{ARuns}(A) \rightarrow \text{ARuns}(B) \) is \textit{word preserving} if \( f(\rho) \) and \( \rho \) are runs over the same infinite word.

**Theorem**

1. \( L(A) \subseteq L(B) \) iff there is a word preserving \( f : \text{ARuns}(A) \rightarrow \text{ARuns}(B) \)
2. \( B \) cont. simulates \( A \) iff there is a \textit{continuous} word preserving \( f : \text{ARuns}(A) \rightarrow \text{ARuns}(B) \)
Consequences

- continuous simulation is a transitive relation
  - continuity is preserved by function composition

- continuous simulation is decidable in 2-EXPTIME
  - follows from [Holtmann, Kaiser and Thomas, FSTTCS 2010]

- on LTS (without fairness), the buffer can be bounded
  - for a LTS $S$, traces($S$) is a compact space
  - on such spaces, continuous functions $\equiv$ Lipschitz functions
  - Lipschitz function $\equiv$ buffer boundedness
**Exact complexity**

**Theorem**

Continuous simulation is EXPTIME-complete.

**Decidability**

- $w \sim_A w'$ if $w, w'$ have the same “transition profile”
- $3|A|^2$ equivalence classes
- cannot abstract buffer exact content by equiv class ...
- first introduce an equivalent game with “lassos”
- end with parity game of exponential size and index 3.
EXPTIME-hardness

Tiling game

- a set of tiles, among which one called “target”
- on every round
  - Starter chooses the first tile in the row

\[ n=4 \]
EXPTIME-hardness

Tiling game

- a set of tiles, among which one called “target”
- on every round
  - Starter chooses the first tile in the row
  - Completer chooses the \( n - 1 \) other tiles of the row
EXPTIME-hardness

Tiling game

- a set of tiles, among which one called “target”
- on every round
  - Starter chooses the first tile in the row
  - Completer chooses the $n - 1$ other tiles of the row

$n=4$
EXPTIME-hardness

Tiling game

- a set of tiles, among which one called “target”
- on every round
  - Starter chooses the first tile in the row
  - Completer chooses the $n-1$ other tiles of the row

$n=4$
EXPTIME-hardness

Tiling game

- a set of tiles, among which one called “target”
- on every round
  - Starter chooses the first tile in the row
  - Completer chooses the \( n - 1 \) other tiles of the row

Completer wins if she eventually puts the target tile.
EXPTIME-hardness (2)

Given a tiling game $G$, define $A_G$ and $B_G$ such that

Starter wins $G$ iff Duplicator wins $\text{contsim}(A_G, B_g)$.

- alphabet = set of tiles + \{0, 1\}
- Spoiler is forced to play a word of the form
  
  \[
  0 \quad 1 \quad 1 \quad \ldots \quad 1 \quad 0 \quad 0 \quad \ldots
  \]

- Duplicator forces Spoiler’s first tile by its position
- Spoiler forces Duplicator to play by repeating a row
Look-ahead simulation

in every round:

1. Spoiler chooses $a \in \Sigma$, and adds it to the buffer
2. Spoiler moves his pebble along an $a$ transition
3. Duplicator can decide to
   - either skip her turn
   - or flush the entire buffer, get $a_1, \ldots, a_k$, and move along $a_1, \ldots, a_k$ transitions
4. winning condition as before
Examples

Duplicator wins

Spoiler wins
PSPACE hardness

**Theorem**

Look-ahead fair simulation is PSPACE hard.

**Proof:**

- reducing from language inclusion for NFA
- for an NFA \(A\), define \(A'\) such that \(L(A') = L(A)\).\(
\#^\infty\) adding an extra state
- then \(L(A) \subseteq L(B)\) iff \(B'\) fairly simulates \(A'\) with look-ahead
**Theorem**

Look-ahead fair simulation is PSPACE hard.

Proof:
- reducing from language inclusion for NFA
- for an NFA $A$, define $A'$ such that $L(A') = L(A)$. $\#^\omega$ adding an extra state
- then $L(A) \subseteq L(B)$ iff $B'$ fairly simulates $A'$ with look-ahead

**Theorem**

Look-ahead unfair simulation is PSPACE hard.

Proof: more involved, again reducing from a tiling problem.
PSPACE upper bound

Theorem
Look-ahead fair simulation is in PSPACE.

Proof:
- define quotient game along the same lines as before
- still get parity game $G$ of exponential size
- however, only $O(|A| \cdot |B|)$ positions for Spoiler in $G$
- solve the game without generating it entirely
A word on automata minimization

Other winning condition

- fair simulation is not good for quotienting
- delayed simulation is [Etessami, Wilke, and Schuller]
- direct simulation is good for pruning

Same holds for continuous/look-ahead counterparts.

Look-ahead simulation is not transitive

- counter-example for multi-letters due to Clemente and Mayr
- carry over look-ahead simulations
- consequence: quotienting is not idempotent, can be repeated
Conclusion

Contribution

- we introduced two new simulations
- we established their decidability and exact complexity
- continuous simulation is mathematically appealing

Perspectives

- high complexities $\Rightarrow$ probably not suitable for NBA
- interesting for timed automata?