Origins of bisimulation

1. Concurrency [Hennessy, Milner, Park]

   Language equivalence is not appropriate for “interacting automata” because it is not a congruence (w.r.t. natural operations)

2. Modal logic [van Benthem]

   Modal logic is a fragment of first-order logic (over transition graphs). Is there an independent characterisation of which fragment? Bisimulation invariance is key notion.

   \( \Phi(x) \) is bisim invariant iff if \( T \models \Phi[s] \) and \( s \sim t \) then \( T \models \Phi[t] \)
Generalizing

- Modal $\mu$-calculus (modal logic + least and greatest fixed points) is “mother-of-all-temporal-logics”, contains LTL, CTL, CTL*, PDL, ...

- Modal $\mu$-calculus is a fragment of monadic second-order logic (over transition graphs). Which fragment?

- Answer again is the bisimulation invariant fragment (Janin and Walukiewicz, 1996)

$\Phi(x)$ is bisim invariant iff if $T \models \Phi[s]$ and $s \sim t$ then $T \models \Phi[t]
PhD Problem 1

- **Satisfiability Problem**: “Given a modal $\mu$-calculus formula $\Phi$, is $\Phi$ satisfiable?” is EXPTIME-complete.

- **Tree Model Property**: If $\Phi$ is satisfiable then $\Phi$ is satisfiable in a transition system/Kripke model that is a (regular) infinite tree.

- **Finite Model Property**: If $\Phi$ is satisfiable then $\Phi$ is satisfiable in a finite transition system/Kripke model.

- **Model Checking Problem**: “Given a finite model, a state $E$ and closed $\Phi$, does $E \models \Phi$?” Its exact complexity is a long standing OPEN problem. Best known upper bound is $\text{NP} \cap \text{co-NP}$.
PhD Problem 2

1. Consider the restricted process language where $I$ is finite

$$E ::= P \mid \sum\{a_i.E_i : i \in I\} \mid E_1 \mid E_2 \mid E \setminus \{a\}$$

A (closed) process, a finite family $\{P_i \overset{\text{def}}{=} E_i : 1 \leq i \leq n\}$ of definitions, where all the process names in each $E_i$ belong to the set $\{P_1, \ldots, P_n\}$.

“Turing powerful” (simulate Turing machines)

2. However if we disallow restriction

$$E ::= P \mid \sum\{a_i.E_i : i \in I\} \mid E_1 \mid E_2$$

Bisimulation equivalence is decidable for this fragment. OPEN QUESTION: is observable bisimulation equivalence decidable?
Hardest PhD Problem

Contrasts between language equivalence and bisimulation equivalence

For finite state transition graphs of size $n$

- Deciding language equivalence is PSPACE-complete. (If the alphabet is singleton then the problem is co-NP complete.)

- Deciding bisimulation equivalence is P-complete. (Naive algorithm is $O(n^2)$)

Show language equivalence isn’t definable within first-order logic with fixed points unlike bisimulation equivalence.
Language Theory and Infinite Graphs

- Usual to think of grammars/automata as word generators

- Alternative process calculus view: generators of labelled graphs

- Is this merely cosmetic?

- It allows one to consider other questions.
Pushdown automata

A pushdown automaton, PDA, consists of

- A finite set of states $P$
- A finite set of stack symbols $S$
- A finite alphabet $A$
- A finite set of basic transitions $T$

Basic transition $pX \xrightarrow{a} q\alpha$ where $p, q \in P$, $X \in S$, $a \in A \cup \{\epsilon\}$ and $\alpha \in S^*$
Configurations

- A configuration $p\beta$, $p \in P$ and $\beta \in S^*$

- Transitions of a configuration

  If $pX \xrightarrow{a} q\alpha \in T$ then $pX\delta \xrightarrow{a} q\alpha\delta$
Transition graphs $G(p\beta)$

$P = \{p, q, r\}$, $S = \{X\}$, $A = \{a, b\}$ and $T$ is

\[
\begin{align*}
pX & \xrightarrow{a} pXX \\
pX & \xrightarrow{b} r\epsilon \\
pX & \xrightarrow{b} q\epsilon \\
qX & \xrightarrow{b} q\epsilon \\
rX & \xrightarrow{\epsilon} r\epsilon \\
rX & \xrightarrow{\epsilon} r\epsilon \\
pX & \xrightarrow{a} pXX \\
pXX & \xrightarrow{a} pXX \\
rX & \xrightarrow{\epsilon} rXX \\
rXX & \xrightarrow{\epsilon} rXX \quad : \quad \ldots
\end{align*}
\]

$G(pX)$ is

\[
\begin{align*}
q\epsilon & \xleftarrow{b} qX \\
qX & \xleftarrow{b} qXX \\
qXX & \xleftarrow{b} qXX \\
qXX & \xleftarrow{b} qXX \\
r\epsilon & \xleftarrow{\epsilon} rX \\
rX & \xleftarrow{\epsilon} rXX \\
rXX & \xleftarrow{\epsilon} rXX \\
rXX & \xleftarrow{\epsilon} rXX \quad : \quad \ldots
\end{align*}
\]

cps Midlands Graduate School today
Some results

- **Language equivalence** is undecidable for context-free grammars (and therefore for pushdown automata).

- **Bisimulation equivalence** is decidable for pushdown automata.

- **Pushdown automata** are richer than context-free grammars with respect to bisimulation equivalence.
**Other Questions I**

- Given a transition graph and vertex $\alpha$, can one decide what are the reachable vertices?

  $$\{\beta : \alpha \xrightarrow{w} \beta \text{ for some word } w\}$$

- **Proposition:** The set of reachable vertices of a pushdown automaton is regular.

- Very useful for infinite-state model checking ("Regular model-checking").

- **Active research area:** more general kinds of graph.
Richer Graphs: Schemes

- A scheme is a finite family

\[ F_i x_1^i \ldots x_n^i \overset{\text{def}}{=} t_i, \ 1 \leq i \leq m \]

of definitions where each \( F_i \) is typed and distinct, and each \( t_i \) is of based type and is built from the typed variables, \( x_1^i, \ldots, x_n^i \), basis functions, the \( F_i \)'s and application

- There is also a start configuration \( S \) of base type without free variables.

- The order of a scheme is the highest order of a free variable \( x_j^i \) that occurs within a definition
2nd-Order Example

\[
F x_1 x_2 x_3 \overset{\text{def}}{=} f(F(G x_1)(H x_2)x_3)x_1(x_2(x_3))
\]
\[
G x_1 x_2 \overset{\text{def}}{=} g(x_1(x_2))
\]
\[
H x_1 x_2 \overset{\text{def}}{=} h(x_1(x_2))
\]

with starting configuration \(F g h a\).
More PhD Problems

- Higher-order schema problem: is following decidable? (open for 30 years)

  Given two schemes do they generate the same (infinite) tree?

- The equivalence problem for first-order schemes interreducible to the equivalence checking problem of deterministic pushdown automata.

- Model-checking problem for schemes: given a scheme, does the tree generated by it have a decidable monadic second-order theory? Decidable Ong (2006) using games