Some remarks on well-structured transition systems

Nikolay V. Shilov

Innopolis University

June 13, 2023

Plan for today

- Firstly I'll introduce Well-structured transition systems formally.
- Then I'll present some "old" results on model checking of disjunctive formulas of the Propositional μ-Calculus (suggested be Dexter Kozen) in intuitionistic models based on well-structured transition systems.
- Finally I'll try to explain in terms of well-structured pre-ordered systems how supercompilation works to detect some infinite program runs.

Well-Pre-Orders

Let D be a set.

- A *pre-order* (or *quasi-order*) is a reflexive and transitive binary relation on D (while anti-symmetry *is not assumed*).
- A *well-pre-order* is a pre-order $\preceq$ such that every infinite sequence $d_0, \ldots d_i, \ldots$ in D contains an *ordered* pair (i.e., such that $m < n$ and $d_m \preceq d_n$.

Ideals or Cones

- An *ideal* (or *(upward-)cone*) is an upward closed subset of D, i.e., a set $I \subseteq D$ such that for all $d', d'' \in D$, if $d' \preceq d''$ and $d' \in I$ then $d'' \in I$.
- Every $d \in I$ generates the upward cone $(\uparrow d) \equiv \{e \in D \ : \ d \preceq e\}$.
- For every set $S \subseteq D$ any element $d \in S$ is a *minimal element* of S, if for every element $s \in S$ either $d \preceq s$ or d and s are non-comparable.
- For any subset $S \subseteq D$, the set of its minimal elements is $min(S)$; a *basis* of S is any subset $B \subseteq S$ such that for every $s \in S$ there exists an element $b \in B$ such that $b \preceq s$.

- Well-pre-ordered set is well-founded (i.e., no infinite strictly decreasing sequence); moreover, every infinite sequence contains an infinite non-decreasing subsequence.
- Every S has a finite basis that comprises all minimal elements $min(S)$; in particular, every ideal I has a finite basis $min(I)$, and $I = \bigcup_{d \in min(I)} (\uparrow d)$.
- Every non-decreasing sequence of ideals $I_0 \subseteq \cdots \subseteq I_i \subseteq \ldots$ eventually stabilizes: there is some $k \geq 0$ such that $I_m = I_n$ for all $m, n \geq k$.

Well-pre-ordered transition system

Let Act be any fixed finite alphabet of action symbols.

- A *transition system* (or *Kripke frame*) is a tuple (D, R), where the domain D is any non-empty set of elements that are called *states* (*worlds* respectively), and the *interpretation* R is a total mapping $R : Act \to 2^{D \times D}$.
- A run (in the frame) is a maximal sequence of states $s_0 \ldots s_i s_{i+1} \ldots$ such that for all adjacent states within the sequence $(s_i, s_{i+1}) \in R(a)$ for some $a \in Act$.
- A well-pre-ordered transition system (WPTS) is a triple $(D, \preceq, R)$ such that $(D, \preceq)$ is a well-pre-ordered set and (D, R) is a Kripke frame.

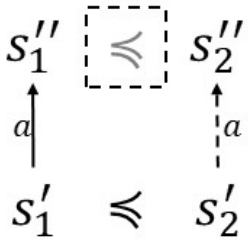
Compatibility of pre-order with transitions

- There are 2 options for *(strong) (future) compatibility* of the well-pre-order $\preceq$ and the interpretation $R(a)$ of an action symbol $a \in Act$ (ref. Fig. 1).
- The adjective *strong* refers to a single step of action $R(a)$ that interprets the corresponding action symbol a.
- The adjective *future* is about states after action(s), i.e., future states, while states before an action are the past states.

Compatibility of pre-order with transitions

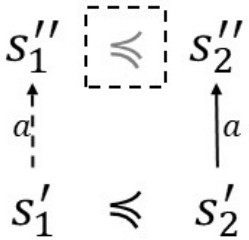
Future Upward – FU

- Logic: $\forall s_1', s_1'', s_2' \exists s_2'' \colon (s_1', s_1'') \in R(a) \,\&\, s_1' \preccurlyeq s_2' \Longrightarrow$
  $\Longrightarrow (s_2', s_2'') \in R(a) \,\&\, s_1'' \preccurlyeq s_2''$
- Algebraic: $(\preccurlyeq)^{-} \circ R(a) \subseteq R(a) \circ (\preccurlyeq)^{-}$



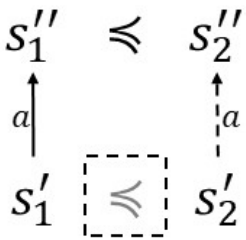
Future Downward – FD

- Logic: $\forall s_1', s_2', s_2'' \exists s_1'' \colon (s_2', s_2'') \in R(a) \,\&\, s_1' \preccurlyeq s_2' \Longrightarrow$
  $\Longrightarrow (s_1', s_1'') \in R(a) \,\&\, s_1'' \preccurlyeq s_2''$
- Algebraic: $(\preccurlyeq) \circ R(a) \subseteq R(a) \circ (\preccurlyeq)$



Past Upward – PU

- Logic: $\forall s_1', s_1'', s_2'' \exists s_2' \colon (s_1', s_1'') \in R(a) \,\&\, s_1'' \preccurlyeq s_2'' \Longrightarrow$
  $\Longrightarrow (s_2', s_2'') \in R(a) \,\&\, s_1' \preccurlyeq s_2'$
- Algebraic: $R(a) \circ (\preccurlyeq) \subseteq (\preccurlyeq) \circ R(a)$



Past Downward – PD

- Logic: $\forall s_1'', s_2', s_2'' \exists s_1' \colon (s_2', s_2'') \in R(a) \,\&\, s_1'' \preccurlyeq s_2'' \Longrightarrow$
  $\Longrightarrow (s_1', s_1'') \in R(a) \,\&\, s_1' \preccurlyeq s_2'$
- Algebraic: $R(a) \circ (\preccurlyeq)^{-} \subseteq (\preccurlyeq)^{-} \circ R(a)$

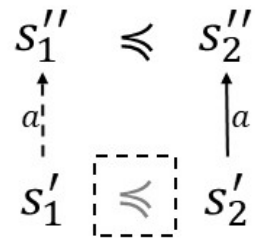


Figure: Pre-order-and-transition compatibility

- An upward compatible well-pre-ordered transition system is called *Well-Structured Transition System* (WSTS).
- Simulation and bisimulation can be defined in terms of compatibility conditions:
  - Future upward compatibility with a binary relation $\preceq$ means that the relation is a *simulation* relation.
  - Future downward compatibility with the *inverse* of a binary relation $\preceq$ means that the *inverse relation* $(\preceq)^{-}$ is a simulation relation.
  - Future upward compatibility with an *equivalence relation* $\simeq$ means that the relation $\simeq$ is a *bisimulation*.

Model Checking Problem

Model checking is an algorithmic problem to find (to compute) semantics of a given formula (of some given logic) in a given model (for this logic).

Theorem

The model checking problem is decidable for disjunctive formulas of the Propositional μ-Calculus in intuitionistic models over well-structured transition systems with decidable well-pre-order tractable pasts.

Theorem explained: pre-order requirements

- The *decidability for the well-pre-order* means that $\preceq$ is decidable as a binary relation on states.
- *Tractable past* means that for any $a \in Act$ the function $\lambda s \in D \ : \ \min\{t \in D \ : \ (t, s) \in R(a)\}$ is computable.

Theorem explained: intuitionistic models

Let Prp be an alphabet of *propositional variables*.

- A *(Kripke) model* (or *labeled transition system*) over a frame $F = (D, R)$ (any well-pre-ordered or well-structured transition system $(D, \preceq, R)$ in particular) is F together with *interpretation* (or *valuation*) $I : Prp \to 2^D$ of individual propositional variables by sets of states.
- Interpretation is said to be *intuitianistic*, if the frame is a well-pre-ordered transition system and $I(p)$ is an upward-cone (i.e., an ideal) with respect to used $\preceq$ for every propositional variable $p \in Prp$.
- A model (labeled transition system) is said to be *intuitianistic*, if has an intuitianistic interpretation.

Theorem explained: syntax of μ-Calculus of D.Kozen (μC)

The syntax of μC consists of formulae:

$$\phi ::= \underline{p} \mid (\neg\phi) \mid (\phi \wedge \phi) \mid \underline{(\phi \vee \phi)} \mid ([a]\phi) \mid \underline{(\langle a \rangle \phi)} \mid (\nu p.\ \phi) \mid \underline{(\mu p.\ \phi)}$$

where

- *Prp* and *Act* are (as above) the alphabets of propositional variables and action symbols,
- metavariables ϕ, p, and a range over formulae, propositional variables, and action symbols respectively.
- The context constraint: no instances of a bound (by μ or ν) propositional variables are in the range of odd number of negations.
- Disjunctive formulae constructs are underlined.

Theorem explained: semantic of μ-Calculus of D.Kozen (μC)

In every model $M = (D, R, I)$ the semantics $M(\phi)$ of any formula ϕ is a subset of the domain D.

- $M(p) = I(p)$, $M(\neg\phi) = D \setminus M(\phi)$,
  $M(\psi \wedge \theta) = M(\psi) \cap M(\theta)$, and $M(\psi \vee \theta) = M(\psi) \cup M(\theta)$
- $M([a]\psi) = \{s \ : \ \text{for every } t \in D, \text{if} (s, t) \in R(a) \text{ then } t \in M(\psi)\}$, and
  $M(\langle a \rangle \psi) = \{s \ : \ \text{for some } t \in D \ (s, t) \in R(a) \text{ and } t \in M(\psi)\} \ = \ (R(a))^{-}(M(\psi))$
  where $(R(a))^{-}$ is the inverse of the binary relation $I(a)$

Theorem explained: semantic of μ-Calculus of D.Kozen (μC)

- $M(\nu p.\ psi) =$ the greatest fix-point of the mapping $\lambda S \subseteq D\ .\ \left(M_{S/p}(\psi)\right)$,
  $M(\mu p.\ psi) =$ the least fix-point of the mapping $\lambda S \subseteq D\ .\ \left(M_{S/p}(\psi)\right)$,
  where $M_{S/p}(\psi)$ is the model that agrees with M everywhere but p, $M_{S/p}(p) = I_{S/p}(p) = S$

- *Metacompilation* was suggested by Valentin F.Turchin in late 1960 . We will consider metacompilation as an approach to program analysis designed to detect some cases of program (infinite) looping basing on either bisimulation or *generalization* and *speculative computations* (to be explained later).
- In this section we assume that the set of states D and interpretations R and I for action symbols Act and propositional variables Prp are fixed; because of this convention, let us write in this section
  - $s \xrightarrow{a} t$ instead $(s, t) \in R(a)$,
  - and $p(s)$ instead $s \in I(p)$.

A toy programming language: syntax

Each program α is a finite set consisting of labeled operators:

- labeled assignment l : *a goto J* where $l \in \mathbb{N}$ is a natural number, $a \in Act$ is an action (the body/action of the operator), and $J \subseteq \mathbb{N}$ is a finite set of natural numbers (jumps, maybe the empty set)
- labeled conditional (choice) l : *if p then J else K* where $l \in \mathbb{N}$ is a natural number, $p \in Prp$ is a condition (the condition/test of the operator), and $J, K \subseteq \mathbb{N}$ is a finite sets of natural numbers (positive-jumps and negative-jumps, maybe the empty set any or both).

The initial label (of α) is the least label that marks any labeled operator in the program; an exit label (of α) is any label that has instance(s) in α but does not mark any labeled operator in α.

A toy programming language: semantic (firings)

A *configuration* (of the program α) is any pair (l, s) where l is a label (i.e., a natural number) and s is a state.

- A *firing of a labeled assignment operator* $(l : a\ goto\ J) \in \alpha$ is any pair of configurations $(l, s)(j, t)$ such that $s \xrightarrow{a} t$ and $j \in J$.
- A *positive firing of a labeled conditional (choice) operator* $(l :\ if\ p\ then\ J\ else\ K) \in \alpha$ is any pair of configurations $(l, s)(j, s)$ such that $p(s)$ and $j \in J$.
- A *negative firing of a labeled conditional (choice) operator* $(l :\ if\ p\ then\ J\ else\ K) \in \alpha$ is any pair of configurations $(l, s)(k, s)$ such that not $p(s)$ but $k \in J$.
- A *firing of a labeled conditional (choice) operator* $(l :\ if\ p\ then\ J\ else\ K) \in \alpha$ is any it's positive or negative firing.

A firing (of α) is any firing of any operator of the program.

A toy programming language: semantic (runs)

A *run* (of α) is any (finite or infinite) sequence of configurations $(l_0, s_0) \dots (l_n, s_n)(l_{n+1}, s_{n+1}) \dots$ such that for any consecutive pair of configurations $(l_n, s_n)(l_{n+1}, s_{n+1}$ within this sequence is a firing of the program α

- An *initial run* is any run starting in the initial label.
- A *terminal run* is any finite run ending in any exit label.
- A *looping run* (or *diverging run*) is any infinite run.
- A *complete run* is either a terminal or a looping run.
- *Execution* is any initial complete run.
- An *executional run* is any run that is a sub-run (i.e., a sub-word) of some execution.

Let us extend the definition of bisimulation from frames to models as follows: for any two states $s \simeq t$, $p(s) \Leftrightarrow p(t)$ for any propositional variable.

Lemma

For any program α, any label l, states s, t, and any bisimulation $\simeq$, if $s \simeq t$ and α has a diverging run from (l, s) than it has a diverging run from (l, t).

- *Generalization* is any binary relation $\overset{gen}{\Longrightarrow}$ on D (i.e., $\overset{gen}{\Longrightarrow} \subseteq D \times D$).
- If $\overset{gen}{\Longrightarrow}$ is a generalization, then for any labels l, k, any states s, t let us write $(l, s) \overset{gen}{\Longrightarrow} (k, t)$ if $l = k$ and $s \overset{gen}{\Longrightarrow} t$.
- Let us say that a generalization $\overset{gen}{\Longrightarrow}$ is *sound* (for our fixed program α) if for any states $s, t, u \in D$ and any labels l, k, $s \overset{gen}{\Longrightarrow} t$ and (*generalized* or *speculative*) firing $(l, t)(k, u)$ together imply that there exists a state $v \in D$ such that $v \overset{gen}{\Longrightarrow} u$ and $(l, s)(k, v)$ is a firing also (*actual firing*).
- It is easy to see that soundness is future downward compatibility from Fig. 1.

Lemma

For any program α, any sound generalization $\stackrel{gen}{\Longrightarrow}$, any label l, any actual state $s \in D$, and a generalized state $t \in D$ such that $s \stackrel{gen}{\Longrightarrow} t$, if there exists a finite speculative run ρ starting and ending in (l, t), then α has an infinite actual run that is generalizable to the infinite speculative run ρ^ω (i.e., the infinite iteration of ρ).

The method on the next slides is based on the above Lemma 3.

Method specification

Precondition: α is a program, (k, r) is its configuration, and *Gen* is a set of sound generalizations.

Invariant: For any configuration (l, s) reachable from (k, r), if the configuration is marked by *loop*, then there exists a looping run starting in (l, s).

Method procedure

Let us construct all runs of α starting in (k, r) in depth-first manner.

- As soon as we detected in any path from (k, r) any two configurations (l, s) and (l, v) on a run that both can be generalized (using any generalization $\stackrel{gen}{\Longrightarrow}$ in *Gen*) to some speculative configuration (l, t), $(l, s) \stackrel{gen}{\Longrightarrow} (l, t)$ and $(l, v) \stackrel{gen}{\Longrightarrow} (l, t)$, such that there exists a speculative run starting and ending in this speculative configuration (l, t), then we may mark both configurations (l, s) and (l, v) in by a special label *loop*.

Roughly speaking, metacompilation is attempting something á la *cross-world predication*: it tries to run speculative computations but then make conclusions about actual ones.