Fixed-point transfer results:

Recall:

- If $(C, \leq) \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} (A, \sqsubseteq)$ is a Galois connection and $\alpha \circ f = f^\# \circ \alpha$, then $\alpha(\mathrm{lfp}.f) = \mathrm{lfp}.f^\#$.

- In the case of wpos, we have $\gamma \circ \alpha = \mathrm{id}$.

Putting this together, we get:

Theorem (Exact Fixed-Point Transfer):

Let $(C, \leq) \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} (A, \sqsubseteq)$ be a Galois connection with $\gamma \circ \alpha = \mathrm{id}$.
Let $f^\# = \alpha \circ f \circ \gamma$.
Then $\mathrm{lfp}.f = \gamma(\mathrm{lfp}.f^\#)$.

Proof:

Since $f^\# = \alpha \circ f \circ \gamma$,
we have $f^\# \circ \alpha = \alpha \circ f \circ \gamma \circ \alpha$ ($\gamma \circ \alpha = \mathrm{id}$)
$= \alpha \circ f$.
Hence, $\alpha(\mathrm{lfp}.f) = \mathrm{lfp}.f^\#$.
Hence, $\gamma\alpha(\mathrm{lfp}.f) = \gamma(\mathrm{lfp}.f^\#)$.

□

Definition:

- A domain $(\Sigma, COM, [\![-]\!])$ is <u>well-structured</u>, if there is a partial order $\le \subseteq \Sigma \times \Sigma$ so that
  - $\le$ is a wpo and
  - $\le$ is a simulation wrt. $[\![com]\!]$, for all $com \in COM$:

$$\left(\begin{array}{c} t_1 \\ \vee\!| \\ s_1 \xrightarrow{[\![com]\!]} s_2 \end{array} \right) \Rightarrow \exists t_2.\ \begin{array}{c} t_1 \longrightarrow t_2 \\ \qquad\ \ \vee\!| \\ \qquad\ \ s_2. \end{array}$$

(In the deterministic setting, this is monotonicity.)

- We also call programs in $W(COM)$ well-structured.

Lemma:

If $\le \subseteq \Sigma \times \Sigma$ is a simulation wrt. $[\![com]\!]$ f.a. $com \in COM$,

then $\le \subseteq \Gamma \times \Gamma$ is a simulation wrt. $\rightarrow$,

where $(c_1, s_1) \le (c_2, s_2)$, if $c_1 = c_2$ and $s_1 \le s_2$.

(It also wpo lifts if we restrict ourselves to the control states in the given program.)

Lemma:

Let $\le \subseteq \Gamma \times \Gamma$ and $\rightarrow \subseteq \Gamma \times \Gamma$.

Then $\le$ is a simulation wrt. $\rightarrow$

iff $\rightarrow U$ is upward-closed, for all upward-closed $U \subseteq \Gamma$.

Proof:

"$\Rightarrow$" Consider an upward-closed set $U \subseteq \Gamma$.

Consider $s \in \rightarrow U$ and $x \in \Gamma$ with $x \geq s$.

Since $s \in \rightarrow \mathcal{U}$, there is $u \in \mathcal{U}$ so that $s \rightarrow u$.

By the simulation property,

there is $y \in T$ with $x \rightarrow y$ and $y \geq u$.

Since $\mathcal{U}$ is upward-closed, $u \in \mathcal{U}$, and $y \geq u$,

we have $y \in \mathcal{U}$.

Hence, $x \in \rightarrow \mathcal{U}$.

"$\Leftarrow$" Consider $x \geq s$ and $s \rightarrow u$.

The set $\{u\}\uparrow$ is upward-closed.

This means $\rightarrow(\{u\}\uparrow)$ is upward-closed by assumption.

By definition, $s \in \rightarrow(\{u\}\uparrow)$.

Since $x \geq s$ and $\rightarrow(\{u\}\uparrow)$ is upward-closed,

also $x \in \rightarrow(\{u\}\uparrow)$.

By definition, there is $y \in \{u\}\uparrow$ with $x \rightarrow y$.

Since $y \in \{u\}\uparrow$, we have $y \geq u$. $\square$

<u>Comment</u>:

What we have just shown is that the (backwards) transition relation does not leave the domain (upward-closed sets) that we can represent precisely.

With this lemma and the exact fixed point transfo,
we can compute backwards reachability
of upward-closed sets.

<u>Definition:</u>

Let $Bad \subseteq T$ be upward-closed.
Let $\leq$ be a simulation ord. $\rightarrow \subseteq T \times T$.
Then
$$pred_{Bad} : \mathbb{P}(T) \longrightarrow \mathbb{P}(T)$$
$$X \longmapsto Bad \cup \rightarrow X$$

yields an upward-closed set
when given an upward-closed set,
$$\boxed{pred_{Bad} : \mathbb{P}_{\uparrow}(T) \longrightarrow \mathbb{P}_{\uparrow}(T)}$$

With the exact fixed-point transfo result,
all we need to do to compute
$$lfp.\, pred_{Bad}$$
is to find a best abstract transformer
$$pred_{Bad}^{\#} = \alpha \circ pred_{Bad} \circ \gamma$$
$$= (min\, -) \circ pred_{Bad} \circ (-\uparrow).$$

This only means we need to be able
to compute the minimal elements
that represent the predecessors of minimal elements
(the upward-closed sets represented by them).

This is usually easy.

Theorem (Abdulla '96):

Under mild computability assumptions
reachability of upward-closed sets
(given by minimal elements)
in well-structured while-programs is decidable.

Remark:

The importance of the result stems from the fact
that sets of bad configurations
tend to be upward-closed.
A concurrent program for example is broken
no matter whether 2, 3, 4 or more
threads are in the critical section.

Assume programs operate over a finite set of Boolean variables $Vars$.

This means the set of states is the set of variable assignments

$$\Sigma := Vars \to \mathbb{B}.$$

Rather than denoting assignments explicitly, we use Boolean formulas over $Vars$, $BF(Vars)$, as the abstract domain.

Lemma:

$$(\mathbb{P}(\Sigma), \subseteq) \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} (BF(Vars)/_{\equiv}, \vdash)$$

With $\gamma(F) := \{\sigma \in \Sigma \mid \sigma(F) = true\}$

$$\alpha(S) := \bigvee_{\sigma \in S} F_\sigma$$

is a Galois connection with $\gamma(\alpha(S)) = S$.

Example: $\alpha(\{(x = true, y = false), (x = false, y = false)\})$

$$= (x \wedge \neg y) \vee (\neg x \wedge \neg y)$$

$$\equiv \neg y.$$

In this setting, also the semantics of commands $[\![com]\!] \in \Sigma \times \Sigma$ can be given as a formula $F_{com}(Vars, Vars')$ with

$$\alpha \circ [\![com]\!] \circ \gamma = F_{com}.$$

The representation via $\mathcal{BF}(Vars)$ and $\mathcal{F}(Vars, Vars')$ is often called symbolic.

As for well-structured programs, we generalize the symbolic representation to configurations and obtain a sound and complete verification method.

Problem: How to analyze a program when (ACC) fails?

Solution: Over-approximate the join in the fixed-point computation.

Definition:

Let $(L, \sqsubseteq)$ be a lattice and $f: L \to L$ monotonic.

Then $\nabla : L \times L \to L$ is a widening operator, if

- $l_1, l_2 \sqsubseteq l_1 \nabla l_2$ f.a. $l_1, l_2 \in L$.
- for all chains $(l_i)_{i \in \mathbb{N}}$ the corresponding chain $(l_i^\nabla)_{i \in \mathbb{N}}$ stabilizes, with

Over-approximate the join.

$$l_0^\nabla := \bot$$
$$l_{i+1}^\nabla := l_i^\nabla, \text{ if } f(l_i^\nabla) \sqsubseteq l_i^\nabla,$$
$$l_{i+1}^\nabla := l_i^\nabla \nabla f(l_i^\nabla).$$

Example: $[0,0], [0,1], [0,2], [0,3], [0,4], \ldots$

Truncate. $[0,0], [0,1], [0,2], [0,\infty], [0,\infty], \ldots$

Note:

- When using widening, we approximate the program semantics while computing it.
- When using an abstract domain and an abstract semantics, we drop information in the beginning and in a controlled way and then try to be precise.

Note:

Simulation relations are also useful to compare the behavior of transition systems. The existence of a simulation shows that the branching behavior in one transition system is less constraining than in another. Indeed, on automata, the existence of a simulation implies language inclusion

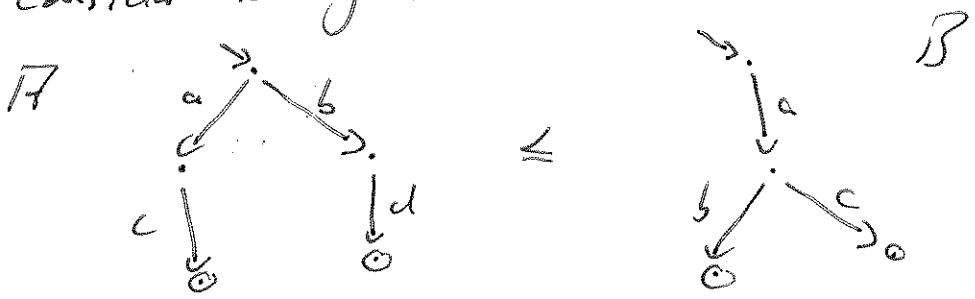
$$A \leq B \Rightarrow \mathcal{L}(A) \subseteq \mathcal{L}(B)$$

($\leq$: exists a simulation relation)

The advantage over language inclusion is that the existence of a simulation can be checked in polynomial time whereas language inclusion is PSPACE-complete.

Example:

Consider two finite automata



In automaton B, the branching behavior is less constraining:

one can still opt for b and c after having seen the a letter.

In this context, the definition of simulation relation is this.

Definition:

Consider finite automata $A = (States_A, \rightarrow_A, init_A, final_A)$, B over Alphabet Σ.

A relation $S \subseteq States_A \times States_B$ is a simulation, if

for all $(s_A, s_B) \in S$: $s_A \in final_A$ iff $s_B \in final_B$

and for all transitions $s_A \xrightarrow{\lambda}_A s_A'$

there is a transition $s_B \xrightarrow{\lambda}_B s_B'$

so that $(s_A', s_B') \in S$ holds.

We write $A \leq B$ if there is a simulation relation S between A and B that relates the initial states, $(init_A, init_B) \in S$.

Lemma:

It can be checked in time polynomial in $|A| + |B|$ whether $A \leq B$ holds.

Proof:

One can show that simulation relations form a complete lattice.

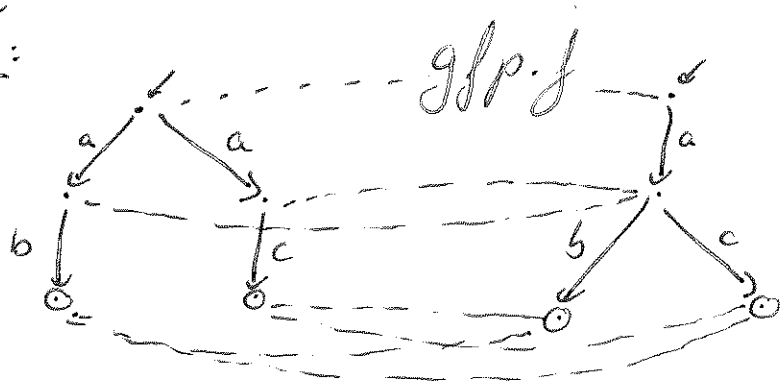
Hence, to check whether one exists, we can just compute the largest one as a greatest fixed point.

Consider

$$f : \mathbb{P}(States_A \times States_B) \longrightarrow \mathbb{P}(States_A \times States_B)$$

$$S \longmapsto \{(s_A, s_B) \in S \mid \forall\, s_A \xrightarrow{a} s_A' \;\; \exists\, s_B \xrightarrow{a} s_B'.\ (s_A', s_B') \in S\}$$

Example:



Lemma:

- $gfp.f$ is the largest simulation relation. Hence, $A \leq B$ holds iff $(init_A, init_B) \in gfp.f$.
- Every fixed point of f is a simulation relation and vice versa.

Remark:

In the context of our lecture,
one should be able to show
that

$$TS(c) \leq TS(c)^{\#}$$

holds, even when we make
visible the commands on transitions.
Here, $TS(c)$ is the transition system
over configurations T defined by $\rightarrow$,
and $TS(c)^{\#}$ is the transition system
over configurations $T^{\#}$ defined by $\rightarrow^{\#}$.
A candidate for a simulation relation
would be

$$S = \{ ((c,\sigma), (c,a)) \mid \alpha(\sigma) \in a \}.$$

Remark:

When a candidate for a simulation relation is given,
it is very easy to check automatically (with SMT solvers)
whether it is a simulation indeed.
Just compute one transition (though $\forall\exists$ is difficult)
and check the result.