Abstract Interpretation from Büchi Automata

Martin Hofmann, Wei Chen, LICS '14.

Goal:
- Model checking functional programs P against policy automaton A:

$$L(P) \subseteq L(A).$$

- Policy automata generate finite and infinite words,

$$L(A) \subseteq \Sigma^{\leq \omega} := \Sigma^\omega \cup \Sigma^*$$

Approach:
- Define a finite lattice M and a Galois connection

$$\mathcal{P}(\Sigma^{\leq\omega}) \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} M.$$

Recall that a Galois connection requires

$$\gamma(\alpha(L)) \supseteq L$$

$$\alpha(\gamma(m)) \leq m.$$

Alternatively, require

$$\alpha(L) \leq m \quad \text{iff} \quad L \in \gamma(m).$$

- The abstraction is faithful wrt. inclusion, more precisely, we will have

$$\gamma(\alpha(L(A))) = L(A),$$

and hence

$$L \subseteq L(A) \quad \text{iff} \quad \alpha(L) \subseteq \alpha(L(A))$$

for all $L \subseteq \Sigma^{\leq\omega}$, in particular $L(P)$.

- The abstraction is chosen such that the language-theoretic operations used in the construction of $L(P)$ can be applied on the level of the abstraction:

$$\cup^\#, \ \cdot^\#, \ *^\#, \ \omega^\#.$$

With this, we can give a type system to compute $\alpha(L(P))$.

- Technically, the existence of these abstract operations follows from the Galois connection.

How?

F ... operation of interest

$$\begin{array}{ccc} L & \xrightarrow{\quad F \quad} & F(L) \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ \alpha(L) & \dashrightarrow & \alpha(F(L)) \end{array}$$

$F^\#$ ... we know the desired value:

$$F^\#(\alpha(L)) = \alpha(F(L)),$$

- Note that if we have

$$F^\# \circ \alpha = \alpha \circ F,$$

then we obtain

$$\alpha(\mathrm{lfp}\ F) = \mathrm{lfp}\ F^\#.$$

- Difficult: ω-iteration

$\alpha(\mathbb{P}(\Sigma^*))$ ... substantive containing abstractions of languages of finite words

$$\omega^\# : M_* \to M$$

with $\alpha(L^\omega) = \alpha(L)^{\omega^\#}$

1. Extended Büchi Automata

Idea: Büchi automata that accept finite and infinite words.

Definition:

An extended Büchi automaton (EBA) is a tuple

$$(Q, \Sigma, \delta, q_0, F)$$

with $\delta: Q \times \Sigma \to \mathcal{P}(Q)$.

The behavior is as expected:

$$L(A) = \underbrace{L^*(A)}_{\text{see } A \text{ as NFA}} \cup \underbrace{L^\omega(A)}_{\text{see } A \text{ as NBA}}$$

$$= \{ w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \}$$

$$\cup \{ w \in \Sigma^\omega \mid q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots \text{ visits } \infty\text{-many final states } \}.$$

Note:

- If $w \in L(A)$ is an ω-word, then all (infinitely many) finite prefixes of w also belong to $L(A)$.
- Hence, EBA cannot accept all union of regular and ω-regular languages.

Philosophy:

- Terminating computations of P $\rightsquigarrow$ issue a special event, say $\$$.

- The finite computations not ending in $\Downarrow$ stem from non-terminating computations that no longer show visible behavior. We let them eventually only show τ, infinitely often. If now every state has a τ-loop, acceptance of such words is covered by the Büchi condition.

- The following theorem gives an understanding of the expressiveness of EBA.

Theorem:

Let $L_1 \subseteq \Sigma^*$ be regular and

$L_2 \subseteq (\Sigma \cup \{\tau\})^\omega$ be ω-regular.

Assume that whenever $w \in L_1$ and

$w' \in \underbrace{w \sqcup \{\tau\}^\omega}$

infinitely many τ only at the end of w

then $w' \in L_2$.

Then there is an extended Büchi automaton for

$L_1 \cdot \Downarrow \cup \underbrace{L_2^\tau}$

words in L_2 with τ removed.

1.1 Equivalence Classes

Fix an EBA $A = (Q, \Sigma, \delta, q_0, F)$.

Recall that for $u, v \in \Sigma^+$, we have

$u \sim_A v$, if $\forall q, q' \in Q.$ $q \xrightarrow{u} q'$ iff $q \xrightarrow{v} q'$

$\wedge$ $q \xrightarrow{u}_F q'$ iff $q \xrightarrow{v}_F q'$.

We extend this to Σ^* by setting $\varepsilon \sim_A \varepsilon$.

We write $[u]_{\sim_A}$ or just $[u]$ for the equivalence class of u.

We write $\Sigma^*/\sim_A$ for the quotient set.

A class $[u]$ can be represented by

the two relations on states it induces:

$\{(q, q') \mid q \xrightarrow{u} q'\}$ and $\{(q, q') \mid q \xrightarrow{u}_F q'\}$.

Lemma:

If A has n states,

there are at most $3^{n^2}+1$ equivalence classes.

Proof:

See the two relations on states as functions

$Q \times Q \longrightarrow \{$ no path, non-accepting path, accepting path $\}$. □

Lemma:

$(\Sigma^*/\sim, \odot, [\varepsilon])$ is a monoid

with $[u] \odot [v] := \underbrace{[uv]}_{\text{concatenation plus quotient}}$.

For well-definedness, check that $u_1 \sim_A u_2$ and $v_1 \sim_A v_2$ implies $u_1 v_1 \sim_A u_2 v_2$.

Lemma:

Let A be an LBA.

(1) The classes of $\Sigma^*/_{\sim_A}$ are regular languages.

(2) For all classes $C \in \Sigma^*/_{\sim_A}$ with $C \cap L(A) \neq \emptyset$
we have $C \subseteq L(A)$.

(3) For all classes $C, D \in \Sigma^*/_{\sim_A}$ with $C.D^\omega \cap L(A) \neq \emptyset$
we have $C.D^\omega \subseteq L(A)$.

(4) For all $w \in \Sigma^{\leq \omega}$ there are $C, D \in \Sigma^*/_{\sim_A}$
with $w \in C.D^\omega$, $CD = C$, and $DD = D$.

Note:

By (3) and (4), sets $C.D^\omega$ (with $CD = C$ and $DD = D$)
behave almost like classes.

We may, however, have

$$C.D^\omega \cap U.V^\omega \neq \emptyset$$

while $C.D^\omega \neq U.V^\omega$.

2. Büchi Abstraction

Idea: The above lemma shows that
we can use pairs of classes (C, D) with $CD = C$
and $DD = D$
to represent languages from $\Sigma^{\leq \omega}$.

Problem: Have to close sets of pairs under overlaps.

Definition:

- A pair (C, D) with $C.D = C$ and $D.D = D$ is called a patch.
- We define the extent of a patch as

$$\gamma(C,D) := C.D^\omega := \{w_0.w_1.w_2... \in \Sigma^{\leq\omega} \mid w_0 \in C \text{ and } w_i \in D \text{ f.a. } i>0\}.$$

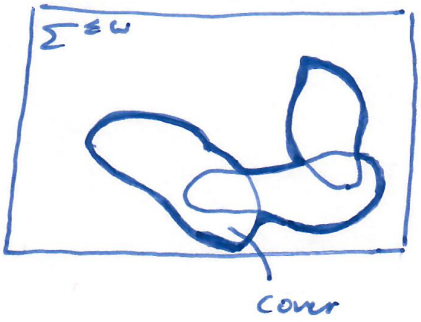
Given a set of patches V the extent is

$$\gamma(V) := \bigcup_{(C,D)\in V} \gamma(C,D).$$

- A set of patches V is closed, if

$$\gamma(C,D) \cap \gamma(V) \neq \emptyset \text{ implies } (C,D) \in V.$$

Illustration:



We call a closed set of patches a cover.

Given a set of patches V, its closure $cl(V)$ is the $\subseteq$-least cover containing V.

- A patch (C,D) is finitary, if $D = [E]$ and thus $\gamma(C,D) \in \Sigma^*$.

Lemma:

(1) $\gamma(C,D) \cap \Sigma^* \neq \emptyset$ if $D = [E]$.

(2) If $D = [E]$, then $\gamma(C,D) \in \Sigma^*$.

(3) $\gamma(C_1, [\varepsilon]) \cap \gamma(C_2, [\varepsilon]) \neq \emptyset$ iff $C_1 = C_2$.

Hence, if for all $(C, D) \in \nu$ we have $D = [\varepsilon]$,
then $\nu = cl(\nu)$.

// Finitary patches are covers.

Proof:
No class $\neq [\varepsilon]$ contains ε. □

2.1 The Abstract Lattice

Definition:

- The abstract lattice is $(\mathcal{M}, \subseteq)$
with
$$\mathcal{M} := \{ \nu \in \Sigma^*/_{\sim_A} \times \Sigma^*/_{\sim_A} \mid \nu \text{ a cover} \},$$
$\subseteq$ the usual inclusion on sets.
- We use $\mathcal{M}_*$ for the sublattice consisting of sets of finitary patches.
- The abstraction function
$$\alpha : \mathcal{P}(\Sigma^{\leq \omega}) \longrightarrow \mathcal{M}$$
is defined by
$$\alpha(L) := cl(\{(C, D) \mid \gamma(C, D) \cap L \neq \emptyset\}).$$

// The abstraction $\alpha(L)$ of L is
the $\subseteq$-smallest closed set of patches
the extents of which cover L.

Theorem:

(1) $(\mathcal{M}, \subseteq)$ is a complete lattice.

$(\mathcal{M}_*, \subseteq)$ is isomorphic to $(\mathbb{P}(\Sigma^*/_{\sim_A}), \subseteq)$.

(2) (α, γ) form a Galois connection between $\mathbb{P}(\Sigma^{\leq\omega})$ and $\mathcal{M}$:

$$L \subseteq \gamma(V) \quad \text{iff} \quad \alpha(L) \subseteq V.$$

(3) $\alpha(\gamma(V)) = V$, so we have a Galois insertion.

(4) $\alpha(L_1 \cup L_2) = \alpha(L_1) \cup \alpha(L_2)$.

$\alpha(T) = T$, i.e. $\alpha(\Sigma^{\leq\omega}) = \Sigma^*/_{\sim_A} \times \Sigma^*/_{\sim_A}$.

$\alpha(\bot) = \bot$, i.e. $\alpha(\emptyset) = \emptyset$.

In general, $\alpha(L_1 \cap L_2) \neq \alpha(L_1) \cap \alpha(L_2)$.

(5) Recall that we consider $\sim_A$ induced by EBA A.

We have

$$\gamma(\alpha(L(A))) = L(A).$$

Hence,

$$L \subseteq L(A) \quad \text{iff} \quad \alpha(L) \subseteq \alpha(L(A)),$$

for all $L \subseteq \Sigma^{\leq\omega}$.

Proof:

(1) Covers are closed under $\cup$ and $\cap$.

For the isomorphism, use the above lemma.

(2) • For the Galois connection, assume $L \subseteq \gamma(V)$.

To show $\alpha(L) \subseteq V$ it is enough

to show $(C,D) \in V$ whenever $\gamma(C,D) \cap L \neq \emptyset$.

Why? Because V is closed.

Suppose $\gamma(C,D) \cap L \neq \emptyset$.

As $L \subseteq \gamma(V)$ by assumption, we get

$$\gamma(C,D) \cap \gamma(V) \neq \emptyset.$$

As V is closed, we obtain $C(C,D) \in V$.

- The converse follows from

$$L \subseteq \gamma(\alpha(L)) \subseteq \gamma(V),$$

assuming $\alpha(L) \subseteq V$.

(3), (4) ✓

(5) Use $C.D^\omega \cap L(A) \neq \emptyset \Rightarrow C.D^\omega \subseteq L(A)$. □

Lemma:

Let C, A be complete lattices

and $C \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} A$ form a Galois insertion, i.e.

- α, γ both monotone
- $\alpha(c) \leq a$ iff $c \leq \gamma(a)$
- $\alpha(\gamma(a)) = a$.

Let $F: C \to C$ and $F^\sharp: A \to A$

with

$$\alpha \circ F = F^\sharp \circ \alpha.$$

Then $\alpha(lfp\, F) = lfp\, F^\sharp$.

Proof:

- Galois insertion implies $\alpha(\bot_C) = \bot_A$.

  To see this, note that

  $$\bot_C \in \gamma(\bot_A).$$

  Hence,

  $$\alpha(\bot_C) \leq \alpha(\gamma(\bot_A)) = \bot_A.$$

  Hence,

  $$\alpha(\bot_C) = \bot_A.$$

- In a Galois connection, α preserves joins and γ preserves meets.

  Reference: "Galois Connections and Computer Science Applications"
  Melton, Schmidt, Strecker.

Hence,

$$\begin{aligned}
\alpha(\mathrm{lfp}\ F) &\overset{(\text{Kleene})}{=} \alpha\left(\bigsqcup_i F^i(\bot)\right) \\
&\overset{(\alpha \text{ preserves } \sqcup)}{=} \bigsqcup_i \alpha(F^i(\bot)) \\
&\overset{(\text{Assumption})}{=} \bigsqcup_i F^{\#i}(\alpha(\bot)) \\
&\overset{(\text{Above})}{=} \bigsqcup_i F^{\#i}(\bot) \overset{(\text{Kleene})}{=} \mathrm{lfp}\ F^{\#}. \qquad \square
\end{aligned}$$

2.2 Concatenation and Iteration

Goal: Introduce operators on the abstract domain that track the language-theoretic operations on $\mathcal{P}(\Sigma^{\leq \omega})$.

Definition:

For $\mathcal{U} \in \mathcal{M}_*$ and $\mathcal{V} \in \mathcal{M}$,

define the abstract concatenation

$$\mathcal{U} \cdot^{\#} \mathcal{V} := \alpha(\{(A \sqcap C, D) \mid (A, [E]) \in \mathcal{U}, (C, D) \in \mathcal{V}\}).$$

Theorem:

(1) If $\mathcal{U} \in \mathcal{M}_*$ and $\mathcal{V} \in \mathcal{M}$, then $\mathcal{U} .\# \mathcal{V} \in \mathcal{M}$.

(2) $\alpha(L_1 \cdot L_2) = \alpha(L_1) .\# \alpha(L_2)$.

(3) If $\mathcal{U}, \mathcal{V} \in \mathcal{M}_*$, then $\mathcal{U} .\# \mathcal{V} \in \mathcal{M}_*$

(4) $.\#$ is monotonic.

Proof:

(1) Consider $(\overline{AC}, D) \in \mathcal{U} .\# \mathcal{V}$.

We have to show $\overline{ACD} = \overline{AC}$.

As $(C, D) \in \mathcal{V}$, we have $CD = C$.

The equality follows.

(2) "$\subseteq$" By the Galois connection, we have

$$L_1 \subseteq \gamma(\alpha(L_1))$$
$$L_2 \subseteq \gamma(\alpha(L_2)).$$

Hence,

$$L_1 \cdot L_2 \in \gamma(\alpha(L_1)) \cdot \gamma(\alpha(L_2))$$
$$\subseteq \gamma(\alpha(L_1) .\# \alpha(L_2)).$$

The second inclusion follows from the definition of $.\#$ and γ.

Now, by monotonicity:

$$\alpha(L_1 \cdot L_2) \subseteq \alpha(\gamma(\alpha(L_1) .\# \alpha(L_2)))$$
$$= \alpha(L_1) .\# \alpha(L_2),$$

using Galois insertion.

.2" Since $\alpha(L_1.L_2)$ is closed,
it suffices to prove

$$\gamma(\overline{A}C, D) \mid (A,[E]) \in \alpha(L_1), (C,D) \in \alpha(L_2)\} \subseteq \alpha(L_1.L_2).$$

Consider $(\overline{A}C, D) \in \alpha(L_1).^{\#}\alpha(L_2)$.

We have

$\gamma(A) \cap L_1 \neq \emptyset$ as there are no closures in $\mathcal{M}_*$.

If $\gamma(C,D) \cap L_2 \neq \emptyset$, we have

$$\gamma(\overline{A}C, D) \cap L_1.L_2 \neq \emptyset,$$

and we are done.

Otherwise, we can assume $\gamma(C,D) \cap \gamma(C',D') \neq \emptyset$,

and for $(C',D') \in \alpha(L_2)$ we have already shown

$$(\overline{A}C', D') \in \alpha(L_1.L_2).$$

Then $\gamma(\overline{A}C, D) \cap \gamma(\overline{A}C', D') \neq \emptyset$

and thus $(\overline{A}C, D) \in \alpha(L_1.L_2)$ by closure.

(3), (4) ✓ □

Since finite iteration can be defined as a least fixed point of operations that we know how to track on the abstraction, it is easy to define $-^{*\#}$ on $\mathcal{M}_*$ so that

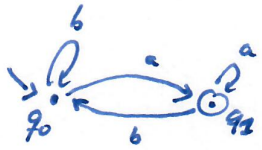
$$\alpha(L)^{*\#} = \alpha(L^*).$$

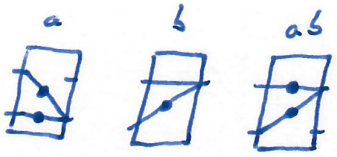
Take

$$\boxed{V^{*\#} := lfp\ \lambda x.\ \alpha(\{\varepsilon\}) \cup^{\#} V.^{\#} x.}$$

The abstraction does not preserve greatest fixed points.

Example:

Let A: 



Let $L = \{a\}$, and thus $\alpha(L) = \{[a], [\varepsilon]\}$.

Then

$\alpha(L)^{\omega^\#}$ should be $\{([a],[a]), ([ab],[ab])\} = \alpha(L^\omega)$.

(intersect)

$=$ $(a+b)^*.a$ $\quad$ $=$ $(a+b)^*.b \setminus b^+$

Yet,

$$gfp\ \lambda x.\ \alpha(L).^\# x$$

also contains $([ab],[b])$.

$=$ b^+

□

The above candidate for $-^{\omega^\#}$ was intuitive,

but not the best abstract transformer.

Definition:

For $U \in \mathcal{A}_\#$, we define the abstract ω-iteration

by

$$U^{\omega^\#} := \alpha(\gamma(U)^\omega).$$

Remark:

Given a Galois connection $C \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} A$

and a function $F: C \to C$,

the best abstract transformer is defined to be

$$\hat{F}: A \to A \quad \text{with} \quad \hat{F} = \alpha \circ F \circ \gamma.$$

-14- This is well-known in abstract interpretation. More in the program analysis course.

The best abstract transformer does not, in general, guarantee

$$\alpha \circ F = \hat{F} \circ \alpha.$$

This is what remains to be shown.

We need Ramsey's theorem.

Lemma:

Let $(L_i)_{i \in I}$ be a family of classes from $\Sigma^*/\sim_A$.

Let

$$P = \prod_{i \in I} L_i \subseteq \Sigma^{\leq \omega}$$ (finite and infinite words $w_1 w_2 \ldots$ with $w_i \in L_i$ for all i).

There are $C, D \in \Sigma^*/\sim_A$ with $CD = C$ and $DD = D$ so that $P \subseteq CD^\omega$.

Proof:

Consider $w \in P$.

We write w as $w_1 w_2 \ldots$ with $w_i \in L_i$.

- If w is finite, then there is an index n so that

$$w_i = \varepsilon \quad \text{and} \quad L_i = [\varepsilon] \quad \text{for all } i > n.$$

Now we choose $C = L_1 \ldots L_{i-1}$ and $D = [\varepsilon]$.
W.l.o.g. $CD = C$ and $DD = D$.

- If w is infinite, we can use Ramsey's theorem to obtain a sequence of indices

$$i_1 < i_2 < \ldots$$

and classes C', D with

$w_1 \dots w_{i_1} \in C'$ and hence $L_1 \dots L_{i_1} = C'$

// If classes overlap, they coincide.

and

$\varepsilon \neq w_{i_1+1} \dots w_{i_2} \in D \neq [\varepsilon]$ and hence $L_{i_1+1} \dots L_{i_2} = D$.

$w_{i_2+1} \dots w_{i_3}$ $L_{i_2+1} \dots L_{i_3}$

$w_{i_3+1} \dots w_{i_4}$ $L_{i_3+1} \dots L_{i_4}$

$\vdots$ $\vdots$

Hence,

$$P = \prod_{i \in I} L_i \subseteq C' D^\omega.$$

- Set $C := C'D$.

Ramsey guarantees $DD = D$.

Hence,

$$C \cdot D = C' \cdot D \cdot D = C' \cdot D = C.$$ □

Theorem:

(1) For all $L \subseteq \Sigma^*$ we have $\alpha(L^\omega) = \alpha(L)^{\omega\#}$.

(2) $-^{\omega\#}$ is monotonic.

Proof:

„$\subseteq$" $L \subseteq \gamma(\alpha(L))$ by the Galois connection,

hence $L^\omega \subseteq \gamma(\alpha(L))^\omega$ by monotonicity of $-^\omega$,

and thus

$$\alpha(L^\omega) \overset{\text{(monotone } \alpha)}{\subseteq} \alpha(\gamma(\alpha(L))^\omega) \overset{\text{(def)}}{=} \alpha(L)^{\omega\#}.$$

.2. Assume $(C,D) \in \alpha(L)^{\omega^*} = \alpha(\gamma(\alpha(L))^\omega)$.

We have to show $(C,D) \in \alpha(L^\omega)$.

Since $\alpha(L^\omega)$ is closed,
it suffices to consider the case that

$$C.D^\omega \cap \gamma(\alpha(L))^\omega \neq \emptyset.$$

- Pick w in the intersection and decompose it into
$$w = w_1 w_2 \dots$$
so that
$$w_i \in \gamma(\alpha(L)) \quad \text{for all } i.$$
Define
$$L_i := [w_i], \text{ and so } w \in \prod L_i.$$

- By the above lemma,
there are C', D' with
$$\prod L_i \in C'.D'^\omega.$$
We thus have
$$w \in C'.D'^\omega \cap C.D^\omega.$$

- Since $w_i \in \gamma(\alpha(L))$,
we must have $[w_i] \cap L = L_i \cap L \neq \emptyset$.
Choose $w_i' \in L_i \cap L$.
Then
$$w_1' w_2' \dots \in L^\omega \cap C'.D'^\omega.$$
Hence,
$$(C', D') \in \alpha(L^\omega).$$

Since

$$C' \cdot D'^{\omega} \cap C \cdot D^{\omega} \neq \emptyset,$$

we conclude

$$(C, D) \in \alpha(L^{\omega}). \qquad \square$$

Remark:

The definition of the abstract ω-iteration
may seem non-constructive
as it goes through the concretization.
Still,

$$\gamma(U)^{\omega} \cap C \cdot D^{\omega}$$

is ω-regular and Büchi non-emptiness
is easy to check.
So $U^{\omega\#}$ is actually computable.

3. A Type System for Recursive Programs

Goal: Check that a given recursive program
adheres to an EBA.

3.1 Recursive Programs

Goal:
- Introduce the programming language:
  first-order, parameterless recursion,
  non-deterministic branching.
- One could introduce state- and data-dependent branching.
  The type system, however, will be oblivious to those.

Definition:

- An expression takes the form

$$e ::= o(a) \mid f \mid e_1 ; e_2 \mid e_1 ? e_2$$

non-deterministic choice

with $a \in \Sigma$ an observable event and $o(a)$ a primitive procedure producing this event. Moreover, f is a procedure identifier from some set ID. Let EXP denote the set of all expressions.

- A program is a function

$$P : F \to EXP,$$

where $F \subseteq ID$ is a finite set of procedure identifiers f each having a defining expression $P(f) = e_f$. We usually omit P and write e_f for the expression defining f. We require well-formedness, all e_f may only contain procedure identifiers from F.

In the following, we define relations

$$e \Downarrow w \text{ and } e \Pi w.$$

Relation $e \Downarrow w$ will mean:

There is a terminating execution of e producing w (finite).

Relation $e \Pi w$ will mean:

There is an execution of e of which w is a finite prefix.

Note:

$$((e \Downarrow w \text{ or } e \Pi w) \text{ and } u \sqsubseteq_{pref} w) \Rightarrow e \Pi u.$$

Definition:

The observed traces are defined

by induction on the structure of expressions

(Structural Operational Semantics, SOS, Plotkin '81):

$$\frac{}{\sigma(a) \Downarrow a} \qquad \frac{}{\sigma(a) \Uparrow a} \qquad \frac{}{e \Uparrow \varepsilon} \qquad \frac{e_f \Downarrow w}{f \Downarrow w} \qquad \frac{e_f \Uparrow w}{f \Uparrow w}$$

$$\frac{e_1 \Downarrow w \quad e_2 \Downarrow u}{e_1 ; e_2 \Downarrow w.u} \qquad \frac{e_1 \Downarrow w \quad e_2 \Uparrow u}{e_1 ; e_2 \Uparrow w.u} \qquad \frac{e_1 \Uparrow w}{e_1 ; e_2 \Uparrow w}$$

$$\frac{e_1 \Downarrow w}{e_1 ? e_2 \Downarrow w} \qquad \frac{e_2 \Downarrow w}{e_1 ? e_2 \Downarrow w} \qquad \frac{e_1 \Uparrow w}{e_1 ? e_2 \Uparrow w} \qquad \frac{e_2 \Uparrow w}{e_1 ? e_2 \Uparrow w} .$$

The infinite behavior requires care.

Consider

$$f = \sigma(a) \quad \text{and} \quad g = (\sigma(a) ; h) ? \sigma(a)$$
$$h = h .$$

The above observed traces are the same for f and g.

The latter, however, has an unproductive infinite recursion.

Idea: Introduce an explicit symbol $\surd$ for idling.

Definition:

- The observed extended trace semantics is defined as above, except that function application is changed to

$$\frac{e_f \Uparrow w}{f \Uparrow \surd . w} .$$

So we work over the alphabet $\Sigma \cup \{\surd\}$.

With this, unproductive recursion leads to $\surd^*$.

- Let $\Theta(w)$ be w with all $\checkmark$ removed.

Definition (Trace Semantics):

Let e be an expression and

$w \in (\Sigma \cup \{\checkmark\})^*$ an extended trace.

We define

$$e \downarrow w := \exists w' \in (\Sigma \cup \{\checkmark\})^*.\ e \Downarrow w' \wedge w = \Theta(w').$$

Since $\Downarrow$ does not introduce $\checkmark$,
this is equivalent to defining

$$e \downarrow w := e \Downarrow w.$$

Moreover, we define

$$e \uparrow w := \exists w' \in (\Sigma \cup \{\checkmark\})^{\omega}.\ \underbrace{(\forall u \sqsubseteq_{pref} w.\ e \Uparrow u)}_{\text{infinite behavior is possible, if all finite prefixes are, König's lemma.}} \wedge w = \Theta(w').$$

Note:

If $e \uparrow w$, then $w \in \Sigma^{\leq \omega}$ and
there is an execution of e on w
that does not terminate.

Example:

$f = \sigma(a); f$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\sigma(a)\Downarrow a} \quad \dfrac{}{f \Uparrow \varepsilon}}{\sigma(a); f \Uparrow a} \quad \parallel \sigma(a); f = e_f}{f \Uparrow a}}{\sigma(a); f \Uparrow a.a}}{f \Uparrow a.a}}{\sigma(a); f \Uparrow a.a.a}$$

$\dfrac{\dfrac{}{\sigma(a)\Downarrow a}}{f \Uparrow a.a.a}$

$\dfrac{\dfrac{\dfrac{}{\sigma(a)\Downarrow a}}{\sigma(a); f \Uparrow a.a.a}}{\dots}$

3.2 Büchi Effects

Goal:
- Over-approximate the traces of an expression e by a pair (U, V) with $U \in \mathcal{M}_*$ and $V \in \mathcal{M}$.
- The following should hold:

  if $e \downarrow w$ then $w \in \gamma(U)$

  if $e \uparrow w$ then $w \in \gamma(V)$.
- The goal is to derive the over-approximation by a type and effect system.

Effect Systems:

Capture the side effects of a computation.
Here, the side effect is the observed extended trace that changes the state of the policy automaton.

Problem: Recursive procedures.

A pair $(A, B) \in \mathcal{V}$ does not track cycles in the invocations:

Solution: Parametrize $\mathcal{V}$ in the above pair (U, V) by information about (the origin of) recursive calls.

Definition:
- Let $\mathcal{X}$ be a finite set of variables.

  A parametrized cover $V(\mathcal{X})$ is an expression

$$\bigcup_{X \in \mathcal{X}}^{\#} (A_X \,.^{\#}\, X) \cup^{\#} B,$$

  where $A_X \in \mathcal{M}_*$ and $B \in \mathcal{M}$.

- A <u>valuation</u> is a function

$$\eta : \mathcal{X} \longrightarrow \mathcal{M}.$$

We use $V(\eta)$ for the element in $\mathcal{M}$ that results from $V(X)$ by replacing each variable X by $\eta(X)$.

- We extend the operations $U.\#$ with $U \in \mathcal{M}_\infty$ and $\cup^\#$ to parameterized covers.

Let $V^i(X) = \bigcup^\#_{X \in \mathcal{X}} (A^i_X .\# X) \cup^\# B^i$, $i = 1, 2$.

Then

$$U .\# V^1(X) := \bigcup^\#_{X \in \mathcal{X}} ((U .\# A^1_X) .\# X) \cup^\# (U .\# B^1)$$

$$V^1(X) \cup^\# V^2(X) := \bigcup^\#_{X \in \mathcal{X}} ((A^1_X \cup^\# A^2_X) .\# X) \cup^\# (B^1 \cup^\# B^2)$$

<u>Definition</u>:

- A <u>Büchi effect</u> is a pair $(U, V(X))$ with $U \in \mathcal{M}_\infty$ and $V(X)$ a parameterized cover.
- An <u>environment</u> Δ binds procedure identifiers f to Büchi effects of the form (U, X), denoted by $f \mathrel{\&} (U, X)$.

If Δ contains $f \mathrel{\&} (U, X)$ and $g \mathrel{\&} (U', X')$, then $X \neq X'$.

So the variables are unique per procedure identifier, and we also denote by X_f the variable for f.

- A typing-judgement takes the form

$$\Delta \vdash e \,\&\, (U, V(X)),$$

where e is an expression
whose procedure identifiers are bound by Δ.

- The derivation relation $\vdash$ for typing judgements is defined by the following rules:

$$(PRIM)\ \frac{}{\Delta \vdash \sigma(a) \,\&\, (\alpha(\{a\}), \emptyset)}$$

$$(SEQ)\ \frac{\Delta \vdash e_1 \,\&\, (U_1, V_1(X)) \qquad \Delta \vdash e_2 \,\&\, (U_2, V_2(X))}{\Delta \vdash e_1 ; e_2 \,\&\, (U_1 \cdot\# U_2,\ V_1(X) \cup\# (U_1 \cdot\# V_2(X)))}$$

$$(IF)\ \frac{\Delta \vdash e_1 \,\&\, (U_1, V_1(X)) \qquad \Delta \vdash e_2 \,\&\, (U_2, V_2(X))}{\Delta \vdash e_1 ? e_2 \,\&\, (U_1 \cup\# U_2,\ V_1(X) \cup\# V_2(X))}$$

$$(CALL\text{-}A)\ \frac{}{\Delta, f \,\&\, (U, X) \vdash f \,\&\, (U, X)}$$

$$(CALL\text{-}B)\ \frac{\Delta, f \,\&\, (U, X) \vdash e_f \,\&\, (U, (A \cdot\# X) \cup\# V(X \setminus \{X\}))}{\Delta \vdash f \,\&\, (U,\ A^{*\#} \cdot\# V(X \setminus \{X\}) \cup\# A^{\omega\#})}$$

Example:

Let $m = \sigma(b) ? \sigma(a); m$.

Note that the observable behavior of m is

$$a^* b \cup a^\omega.$$

We have the following derivation, with $\Delta = m \,\&\, (\alpha(a^* b), X)$:

$$\dfrac{\dfrac{}{\Delta \vdash \sigma(a) \,\&\, (\alpha(\{a\}), \emptyset)}\,(\text{PRIM}) \quad \dfrac{}{\Delta \vdash m \,\&\, (\alpha(a^*b), x)}\,(\text{CALL-A})}{\Delta \vdash \sigma(a); m \,\&\, (\underbrace{\alpha(\{a\}).\#\alpha(a^*b)}_{\alpha(a^+b)}, \alpha(\{a\}).\#x)}\,(\text{SEQ})$$

Moreover, we obtain

$$\dfrac{}{\Delta \vdash \sigma(b) \,\&\, (\alpha(\{\}), \emptyset)}\,(\text{PRIM}) .$$

Combined via (IF), the two derivations yield

$$\Delta \vdash \sigma(b) ? \sigma(a); m \,\&\, (\underbrace{\alpha(\{\}) \cup \#\alpha(a^+b)}_{\alpha(a^*b)}, \alpha(\{a\}).\#x).$$

Then with (CALL-B) we obtain:

$$\vdash m \,\&\, (\alpha(a^*b), \alpha(\{a\})^{\omega\#}).$$

Here, $\alpha(a^*b)$ in the initial environment was cleverly chosen.

Take a least fixed point – more in a moment.

Moreover, note that $\alpha(\{a\})^{\omega\#} = \alpha(a^\omega)$.

Intuition to Rule (CALL-B):

The premise says an infinite trace of e_f
- ↳ will either be captured by $V(X \setminus \{X\})$
- ↳ or begin with a finite prefix A followed by a call to f.

Hence the conclusion says an infinite trace of f
- ↳ will either go through A finitely often and then evolve along $V(X \setminus \{X\})$.
- ↳ or keep doing A forever.

Definition:

- The environment Δ is justified, if

  for all $f \,\&\, (\mathcal{U}, X) \in \Delta$ one has

  $$\Delta \vdash \mathrm{ej} \,\&\, (\mathcal{U}, (\vec{A} \cdot\!\!\# X) \cup\!\!\# V(X \setminus \{X_f\})),$$

  for some $\vec{A} \in \mathcal{M}_\#$ and some $V(X \setminus \{X_f\})$.

This means $\mathcal{U}$ is a fixed point.

- A valuation η satisfies an environment Δ, if

  for all $f \,\&\, (\mathcal{U}, X) \in \Delta$ and all $f \uparrow w$

  we have $w \in \gamma(\eta(X))$.

- We write $\eta \models \Delta$ to mean

  Δ is justified and η satisfies Δ.

Theorem (Soundness):

Consider environment Δ and valuation η with $\eta \models \Delta$.

Whenever $\Delta \vdash e \,\&\, (\mathcal{U}, V(X))$

then $e \downarrow w$ implies $w \in \gamma(\mathcal{U})$

$e \uparrow w$ implies $w \in \gamma(V(\eta))$.

There is also completeness.

Fix for each procedure identifier f the variable X_f.

For $\vec{A} = (A_f)_{f \in F}$ a family of finitary abstractions ($A_f \in \mathcal{M}_\#$),

define the environment $\Delta(\vec{A})$ to contain bindings $f \,\&\, (A_f, X_f)$.

The function bodies e_f define a monotonic operator (typically non-linear)

on $\mathcal{M}_\#^{X_F} = X_F \longrightarrow \mathcal{M}_\#$.

Let $\bar{B}$ be the least fixed point of this operator.
Then $\Delta(\bar{B})$ is justified and we get

$$\Delta(\bar{B}) \vdash e_j \;\&\; (B_j, U_j(X)),$$

using all rules except (CALL-A).
Successive applications of (CALL-B) then yield

$$\vdash f \;\&\; (B_f, \gamma_f).$$

Induction shows

$$B_f = \alpha(\{w \mid f \downarrow w\}) \quad \text{and} \quad \gamma_f = \alpha(\{w \mid f \uparrow w\}).$$

With this argumentation, we obtain

Theorem (Completeness):
The judgements $\vdash f \;\&\; (\alpha(\{w \mid f \downarrow w\}), \alpha(\{w \mid f \uparrow w\}))$
are derivable for all f.

To conclude, in order to check
whether all traces of e
are accepted by A,
derive $\vdash e \;\&\; (B, \gamma)$ and check $B \cup \gamma \subseteq \alpha(L(A))$.

Note:
The problem is EXPTIME-complete
(Bouajjani, Esparza, Maler, CONCUR '97,
Godefroid, TACAS '13).