An introduction to game semantics for programming languages

Aurore Alcolei

LSD Seminar, 31, October 2018

A tiny bit of history

1992 Blass game for linear logic

1993 Fully abstract model for PCF (AJM, HO)

1997 - . . . Abramsky cube and other features

2003 - . . . Game semantics for verification

What semantics?

A programm

```
int et_g(int n, int m)
{
  if (n) return m;
  else return 0;
}
```

How does it computes?
$\rightarrow$ Operational semantics

What does it computes?
$\rightarrow$ Denotational semantics

What semantics?

A programm

```
int et(int n, int m)
{
  if (n)
  { if (m) return 1;
    else return 0;}
  else
  { if (m) return 0;
    else return 0;}
}
```

How does it computes?

→ Operational semantics

What does it computes?

→ Denotational semantics

Basic properties

- Model: $[\![\_]\!]$: programs/types $\rightarrow$ model

$$f : \text{bool} \rightarrow \text{bool} \quad \Longrightarrow \quad [\![f]\!] \in [\![\ \text{bool} \rightarrow \text{bool}\ ]\!]$$

- Correction: (base types)

$$P : \text{bool}\ , P \rightsquigarrow b \quad \Leftrightarrow \quad [\![P]\!] = [\![b]\!]$$

- Definability:

$$\varphi \in [\![\ \text{bool} \rightarrow \text{bool}\ ]\!] \quad \Longrightarrow \exists f : \text{bool} \rightarrow \text{bool}\ , \varphi = [\![f]\!]$$

Obtained by putting constraints on the model.

More properties

- Observational equivalence (any types)

$$P \approxeq Q \ := \ \forall C[\_] : \text{bool} , C[P] \rightsquigarrow b \ \Leftrightarrow \ C[Q] \rightsquigarrow b$$

- Adequacy

$$[\![P]\!] = [\![Q]\!] \quad \Longrightarrow \quad P \approxeq Q$$

- Completeness

$$P \approxeq Q \quad \Longrightarrow \quad [\![P]\!] = [\![Q]\!]$$

- Decidability "=" is decidable in the model.

Adequacy?

A context

```
int main()
{
  int i = 0;
  et(i = 0,i = 1);
  if (i) while (1) {};
  return 0;
}
```

Two programs

```
int et(int n, int m)
{
  if (m)
  { if (n) return 1;
    else return 0;}
  else
  { if (n) return 0;
    else return 0;}
}
```

```
int et(int n, int m)
{
  if (n)
  { if (m) return 1;
    else return 0;}
  else
  { if (m) return 0;
    else return 0;}
}
```

Game semantics

From value to traces

$[\![\tau]\!]$: 2-player games (O,P)
Play: alternating sequences of move, O starts

From function to strategies

$[\![P : \tau]\!]$: strategy over $[\![\tau]\!]$
alternating sequences that respects the game + other restrictions.

The language (a fragment of IA) (1)

- basic types: σ := bool, int, var[bool/int], comm
- constants:
  - n:int, true:bool, false:bool
  - skip:comm, coucou: comm
  - – := – : var[int] $\to$ int $\to$ comm   !– : var[int] $\to$ int
  - if – then – else – : bool $\to \sigma \to \sigma \to \sigma$
  - – ; – : comm $\to$ comm
  - while – do – : bool $\to$ comm $\to$ comm
  - arithmetic/logic operators ($+, \times$, $>$, $<$, ...)
- constructors:
  - fun x $\mapsto$ M[x]
  - new x : var in M[x]
  - M N

The language (a fragment of IA) (2)

An non trivial language ...

- new x in x := 3 ; while (x > 0) do (coucou ; x := !x - 1)
- while true do skip

... with some restriction : function types are restricted to first order

$$\theta := \sigma \mid \sigma \to \theta$$

Roadmap

The Abramsky-Mc Custer game semantics

General game model

- $[\![\text{Types}]\!] \sim$ 2 Player Games
  - a set of moves $\Sigma = \mathcal{Q} + \mathcal{A}$
  - some structure/rules to describe valid plays of the game

- $[\![\text{Programs}]\!] \sim$ Strategies for Player (subsets of plays)

A regular language model (1)

- $[\![\text{Types}]\!] \sim$ 2 Player Games
  - a set of moves $\Sigma = \mathcal{Q} + \mathcal{A}$
  + some structure/rules to describe valid plays of the game

$\longrightarrow$ Plays in $[\![\theta]\!]$ are strings over the alphabet Σ_θ:

- bool : $\mathcal{Q}_{\text{bool}} = \text{q}$, $\mathcal{A}_{\text{bool}} = \{\text{true}, \text{false}\}$
- int : $\mathcal{Q}_{\text{int}} = \text{q}$, $\mathcal{A}_{\text{int}} = \{-n, \ldots, 0, \ldots, n\} = \mathcal{N}$
- comm : $\mathcal{Q}_{\text{comm}} = \{\text{run}\}$, $\mathcal{A}_{\text{comm}} = \{\text{done}\}$
- var[int] : $\mathcal{Q}_{\text{var}} = \{\text{read}, \text{write}(n)\}$, $\mathcal{A}_{\text{var}} = \{n, \text{done}\}$ $(n \in \mathcal{N})$
- function : $\Sigma_{\sigma\to\theta} = \Sigma_\sigma + \Sigma_\theta$

- $[\![\text{Programs}]\!] \sim$ Strategies for Player (subsets of plays)

A regular language model (1)

- $[\![\text{Types}]\!] \sim$ 2 Player Games
  - a set of moves $\Sigma = \mathcal{Q} + \mathcal{A}$
  + some structure/rules to describe valid plays of the game

$\longrightarrow$ Plays in $[\![\theta]\!]$ are strings over the alphabet Σ_θ:

- bool : $\mathcal{Q}_{\text{bool}} = \text{q}$, $\mathcal{A}_{\text{bool}} = \{\text{true}, \text{false}\}$
- int : $\mathcal{Q}_{\text{int}} = \text{q}$, $\mathcal{A}_{\text{int}} = \{-n, \ldots, 0, \ldots, n\} = \mathcal{N}$
- comm : $\mathcal{Q}_{\text{comm}} = \{\text{run}\}$, $\mathcal{A}_{\text{comm}} = \{\text{done}\}$
- var[int] : $\mathcal{Q}_{\text{var}} = \{\text{read}, \text{write}(n)\}$, $\mathcal{A}_{\text{var}} = \{n, \text{done}\}$ $(n \in \mathcal{N})$
- function : $\Sigma_{\sigma \to \theta} = \Sigma_\sigma + \Sigma_\theta$

- $[\![\text{Programs}]\!] \sim$ Strategies for Player (subsets of plays)

$\longrightarrow$ $[\![\Gamma \vdash M : \theta]\!]$ is a regular expression over $\Sigma_\Gamma + \Sigma_\theta$

A regular language model (2)

Interpretation of some constant :

- $[\![\mathsf{skip} : \mathsf{comm}]\!] = \mathsf{run} \cdot \mathsf{done}$
- $[\![\mathsf{true} : \mathsf{bool}]\!] = \mathsf{q} \cdot \mathsf{true}$
  $[\![\mathsf{false} : \mathsf{bool}]\!] = \mathsf{q} \cdot \mathsf{false}$
- $[\![\mathsf{while} - \mathsf{do} - : \mathsf{bool} \rightarrow \mathsf{comm} \rightarrow \mathsf{comm}]\!] =$
  $\mathsf{run}^0 \cdot (\mathsf{q} \cdot \mathsf{true} \cdot \mathsf{run}^1 \cdot \mathsf{done}^1)^* \cdot \mathsf{q} \cdot \mathsf{false} \cdot \mathsf{done}^0$

...

A regular language model (3)

Interpretation for the application : $[\![MN]\!] = ?$

REPRENDRE cette slide avec un exemple concret de composition de langage (puis $[M] \circ [N]^*$)

$M : \theta_1 \to \theta_0,\ N : \theta_1,\ MN : \theta_0$

$$\Sigma_{\theta_1} \xleftarrow{\pi_1} \Sigma_{\theta_1} + \Sigma_{\theta_0} \xrightarrow{\pi_0} \Sigma_{\theta_0}$$

$\pi_i(a^i \cdot w) = a^i \cdot \pi_i(w) \quad \pi_i(a^{1-i} \cdot w) = \pi_i(w)$

$$[\![MN]\!] = \pi_0((\pi_1^{-1}([\![N]\!]))^* \cap [\![M]\!])$$

(still a regular expression)

Examples (1)

$$\begin{aligned}
&[\![\text{while true do } M]\!] \\
&\qquad = \quad [\![(\text{while} - \text{do} \text{ —}) \text{ true } M]\!] \\
&\qquad = \quad \pi_{\mathsf{comm}_0}((\pi^{-1}_{\mathsf{comm}_1}([\![M]\!]))^* \cap [\![(\text{while} - \text{do} \text{ —}) \text{ true}]\!]) \\
&\qquad = \quad \pi_{\mathsf{comm}_0}((\pi^{-1}_{\mathsf{comm}_1}([\![M]\!]))^* \cap \emptyset) \\
&\qquad = \quad \pi_{\mathsf{comm}_0}(\emptyset) \\
&\qquad = \quad \emptyset \quad = \quad [\![\Omega]\!]
\end{aligned}$$

with

$$\begin{aligned}
&[\![(\text{while} - \text{do} \text{ —}) \text{ true}]\!] \\
&\quad = \quad \pi_{\mathsf{comm}_1 \to \mathsf{comm}_0}((\mathsf{run}^0 \cdot (\mathsf{q} \cdot \mathsf{true} \cdot \mathsf{run}^1 \cdot \mathsf{done}^1)^* \cdot \mathsf{q} \cdot \mathsf{false} \cdot \mathsf{done}^0) \\
&\quad \qquad \cap (\pi^{-1}_{\mathsf{bool}}(\mathsf{q} \cdot \mathsf{true}))^*) \\
&\quad = \quad \pi_{\mathsf{comm}_1 \to \mathsf{comm}_0}(\emptyset) \\
&\quad = \quad \emptyset
\end{aligned}$$

Examples (2)

$$
\begin{aligned}
&[\![\text{while false do } M]\!] \\
&\qquad = \quad [\![(\text{while} - \text{do} \text{ —}) \text{ false } M]\!] \\
&\qquad = \quad \pi_{\text{comm}_0}((\pi^{-1}_{\text{comm}_1}([\![M]\!]))^* \cap [\![(\text{while} - \text{do} \text{ —}) \text{ false}]\!]) \\
&\qquad = \quad \pi_{\text{comm}_0}((\pi^{-1}_{\text{comm}_1}([\![M]\!]))^* \cap \text{run}^0 \cdot \text{done}^0)) \\
&\qquad = \quad \pi_{\text{comm}_0}(\text{run}^0 \cdot \text{done}^0) \\
&\qquad = \quad \text{run} \cdot \text{done} \\
&\qquad = \quad [\![\text{skip}]\!]
\end{aligned}
$$

with

$$
\begin{aligned}
&[\![(\text{while} - \text{do} \text{ —}) \text{ false}]\!] \\
&\quad = \quad \pi_{\text{comm}_1 \to \text{comm}_0}((\text{run}^0 \cdot (\text{q} \cdot \text{true} \cdot \text{run}^1 \cdot \text{done}^1)^* \cdot \text{q} \cdot \text{false} \cdot \text{done}^0) \\
&\quad \qquad \cap(\pi^{-1}_{\text{bool}} (\text{q} \cdot \text{false}))^*) \\
&\quad = \quad \pi_{\text{comm}_1 \to \text{comm}_0}(\text{run}^0 \cdot \text{q} \cdot \text{false} \cdot \text{done}^0) \\
&\quad = \quad \text{run}^0 \cdot \text{done}^0
\end{aligned}
$$

It's time to conclude

An effectively computable model which is sound and adequate (in fact it is even more fully abstract ...)

$\rightarrow$ decidability of program equivalence

$\rightarrow$ a semantic suitable for program verification?

It’s time to have some snacks!

Thanks for your attention :)