Sémantique dénotationnelle

... d'un langage impératif jouet,
et conséquences

Rappel

- **Thm**: Les deux propriétés suivantes sont équivalentes:
  - $\rho \vdash c \Rightarrow \rho_\infty$ est dérivable [grand pas]
  - $(c \,.\, \varepsilon, \rho) \rightarrow^* (\varepsilon, \rho_\infty)$ [petits pas]
  - $[\![c]\!]\rho = \rho_\infty \neq \perp$ [dénotationnelle].

Equivalence contextuelle

- **Defn.** $(c, \rho) \cong (c', \rho')$ ssi $\forall$ contexte C,
  $(c \,.\, C, \rho)$ termine $\Leftrightarrow$ $(c' \,.\, C, \rho')$ termine.

- On peut remplacer « termine » par certaines autres propriétés, par exemple « finit par mettre x à 0 »

- Notion d'équivalence purement opérationnelle. Sera utile dans d'autres langages (calculs de processus).

Abstraction complète

- **Thm.** $(c, \rho) \cong (c', \rho')$ ssi $[\![c]\!]\rho = [\![c']\!]\rho'$.
- $\Leftarrow$: par correction et adéquation
- $\Rightarrow$: soit $\rho_\infty = [\![c]\!]\rho \neq \rho'_\infty = [\![c']\!]\rho'$, disons $\rho_\infty \neq \bot$

  … construire C tel que

  $[\![c; C]\!]\rho \neq \bot$ et $[\![c'; C]\!]\rho' = \bot$: une idée?

Plus faibles préconditions

- **Defn.** Propriété P = fonction : $Env \to Bool$.
  $wp(c)(P)$ = propriété Q la moins forte
  t.q. pour tout ρ vérifiant Q, $[\![c]\!]\rho$ vérifie P.

- On peut vérifier que:
  $$wp(c)(P)(\rho) = P([\![c]\!]\rho).$$

- Mais on peut aussi donner une définition récursive de $wp(c)(P)$…

Plus faibles préconditions

- *wp*(c)(*P*)(ρ) = *P*(⟦c⟧ρ).

$$
\begin{aligned}
wp(x := e)(P) &= (\rho \in Env \mapsto P(\rho[x \mapsto [\![e]\!]\,\rho])) \\
wp(\texttt{skip})(P) &= P \\
wp(c_1; c_2)(P) &= wp(c_1)(wp(c_2)(P)) \\
wp(\texttt{if}\, c_1\, \texttt{then}\, c_2\, \texttt{else}\,)(P) &= (\rho \in Env \mapsto \\
&\qquad ([\![e]\!]\,\rho \neq 0 \wedge wp(c_1)(P)(\rho)) \\
&\qquad \vee([\![e]\!]\,\rho = 0 \wedge wp(c_2)(P)(\rho)) \\
wp(\texttt{while}\, e\, \texttt{do}\, c)(P) &= \mathrm{lfp}(\mathrm{Pre}^P_{e,c})
\end{aligned}
$$

où $\mathrm{Pre}^P_{e,c}$:

propriété *Q* ↦ propriété $(\rho \in Env \mapsto$
$([\![e]\!]\,\rho = 0 \wedge P(\rho))$
$\vee([\![e]\!]\,\rho \neq 0 \wedge wp(c)(Q)(\rho)))$

Calcul des wp

- Syntaxe: formules F (dans une logique suffisamment expressive).
- On définit une formule $\langle c \rangle F$ pour chaque commande F qui est la précondition (syntaxique) de c pour assurer F.
- I.e., on veut $[\![\langle c \rangle F]\!] = \mathrm{wp}(c)([\![F]\!])$.

Calcul de $\langle c \rangle F$ [Dijkstra75]

$$
\begin{aligned}
\langle x := e \rangle F &= F[x := e] \\
\langle \mathtt{skip} \rangle F &= F \\
\langle c_1 ; c_2 \rangle F &= \langle c_1 \rangle \langle c_2 \rangle F \\
\langle \mathtt{if}\ e\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2 \rangle F &= (e \dot{\neq} \dot{0} \wedge \langle c_1 \rangle F) \vee (e \doteq \dot{0} \wedge \langle c_2 \rangle F) \\
\langle \mathtt{while}\ e\ \mathtt{do}\ c \rangle F &= \forall A \cdot ((e \doteq \dot{0} \wedge F) \vee (e \dot{\neq} 0 \wedge \langle c \rangle A(\vec{x}))) \Rightarrow A(\vec{x})
\end{aligned}
$$

En pratique, la quantification du second ordre sur *A* est remplacée par quelque chose d'incomplet mais de plus pratique.

Logique de Hoare

[Floyd67, Hoare69]

(ici, pour la correction totale)

Idée:

$\{F\}c\{G\}$

$\Leftrightarrow$

$F \Rightarrow \langle c \rangle G$

$$\frac{}{\{F[x := e]\}\ x := e\ \{F\}}\,(H :=) \qquad \frac{}{\{F\}\ \texttt{skip}\ \{F\}}\,(H\,\texttt{skip})$$

$$\frac{\{F \wedge e \doteq \dot{0}\}\ c_1\ \{F'\} \quad \{F \wedge e \dot{\neq} \dot{0}\}\ c_2\ \{F'\}}{\{F\}\ \texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{F'\}}\,(H\,\texttt{if})$$

$$\frac{\{F\}\ c_1\ \{G\} \quad \{G\}\ c_2\ \{F'\}}{\{F\}\ c_1; c_2\ \{F'\}}\,(H;)$$

$$\frac{\{I \wedge e \dot{\neq} \dot{0} \wedge e' = x\}\ c\ \{I \wedge e' \prec x\}}{\{I\}\ \texttt{while}\ e\ \texttt{do}\ c\ \{I \wedge e \doteq \dot{0}\}}\,(H\,\texttt{while}_t)$$

$$\frac{\{G\}\ c\ \{G'\}}{\{F\}\ c\ \{F'\}}\,(H \Rightarrow)$$
$$\text{si } \mathcal{L} \vdash F \Rightarrow G$$
$$\text{et } \mathcal{L} \vdash G' \Rightarrow F'$$

Passage à la continuation

- On s'aperçoit que, dans $wp(c)(P)$, P peut être de type $Env \to Ans$ plutôt que $Env \to Bool$.
- Traditionnellement on le note alors
  $\kappa : Env \to Ans$
- C'est une **continuation**: dit quoi faire de la valeur obtenue à la fin du calcul.

Passage à la continuation

- Soit *Ans* un dcpo pointé quelconque.

$$
\begin{aligned}
C[\![x := e]\!]\,\kappa\rho &= \kappa(\rho[x \mapsto [\![e]\!]\,\rho]) \\
C[\![\mathtt{skip}]\!]\,\kappa\rho &= \kappa(\rho) \\
C[\![c_1; c_2]\!]\,\kappa\rho &= C[\![c_1]\!]\,(C[\![c_2]\!]\,\kappa)(\rho) \\
C[\![\mathtt{if}\ e\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2]\!]\,\kappa\rho &= \begin{cases} [\![c_1]\!]\,\kappa\rho & \text{si } [\![e]\!]\,\rho \neq 0 \\ [\![c_2]\!]\,\kappa\rho & \text{si } [\![e]\!]\,\rho = 0 \end{cases} \\
C[\![\mathtt{while}\ e\ \mathtt{do}\ c]\!]\,\kappa\rho &= \left(\sup_{n\in\mathbb{N}} Pre^{n}_{e,c}(\kappa)\right)(\rho)
\end{aligned}
$$

dans [*Env* → *Ans*]

$\rho \mapsto \rho$ si $[\![e]\!]\rho = 0$,
$[\![c]\!]\kappa\rho$ sinon

… donne des idées de nouvelles constructions

- (Scheme)
  ⟦`callcc` *k* `in` *c*⟧κρ = ⟦*c*⟧κ(ρ[*k*↦κ])
  ⟦`throw` e *arg*⟧κρ=κ'(⟦*arg*⟧ρ) où ⟦e⟧ρ=κ'
- (Qu'est-ce que ça fait?)
- Exemple plus simple: traitement d'exceptions, voir notes de cours.