Lenguajes y Compiladores
Introducción

Miguel Pagano

15 de marzo de 2023

Repaso de nuestro lenguaje de fórmulas

La necesidad de un estado

Recordemos nuestro lenguaje:

$$
\begin{aligned}
\langle intexp \rangle ::= \; & \langle var \rangle \mid 0 \mid 1 \mid \ldots \\
\mid \; & -\langle intexp \rangle \\
\mid \; & \langle intexp \rangle * \langle intexp \rangle \mid \langle intexp \rangle \; / \; \langle intexp \rangle \mid \langle intexp \rangle \; \% \; \langle intexp \rangle \\
\mid \; & \langle intexp \rangle + \langle intexp \rangle \mid \langle intexp \rangle - \langle intexp \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle assert \rangle ::= \; & \textbf{true} \mid \textbf{false} \\
\mid \; & \langle intexp \rangle \text{ `=' } \langle intexp \rangle \mid \langle intexp \rangle < \langle intexp \rangle \mid \langle intexp \rangle \leqslant \langle intexp \rangle \\
\mid \; & \langle intexp \rangle \neq \langle intexp \rangle \mid \langle intexp \rangle > \langle intexp \rangle \mid \langle intexp \rangle \geqslant \langle intexp \rangle \\
\mid \; & \neg \langle assert \rangle \\
\mid \; & \langle assert \rangle \vee \langle assert \rangle \mid \langle assert \rangle \wedge \langle assert \rangle \\
\mid \; & \langle assert \rangle \Rightarrow \langle assert \rangle \mid \langle assert \rangle \Leftrightarrow \langle assert \rangle \\
\mid \; & \exists \langle var \rangle \; . \; \langle assert \rangle \mid \forall \langle var \rangle \; . \; \langle assert \rangle
\end{aligned}
$$

$\langle var \rangle$ es un no-terminal sin constructores: es un conjunto numerable de "variables".

Ecuaciones semánticas

Para expresiones enteras

Recordemos que $\Sigma = \langle var \rangle \to \mathbb{Z}$.

$$
\begin{aligned}
[\![\_]\!]_{intexp} &: \langle intexp \rangle \to (\Sigma \to \mathbb{Z}) \\
[\![v]\!]_{intexp}\sigma &= \sigma\, v \\
[\![\lfloor k \rfloor]\!]_{intexp}\sigma &= k \\
[\![-e]\!]_{intexp}\sigma &= -([\![e]\!]_{intexp}\sigma) \\
[\![e \oplus e']\!]_{intexp}\sigma &= [\![e]\!]_{intexp}\sigma \oplus [\![e']\!]_{intexp}\sigma
\end{aligned}
$$

Ecuaciones semánticas

Para expresiones booleanas

$$
\begin{aligned}
[\![\_]\!]_{assert} &: \langle assert \rangle \to (\Sigma \to \{0,1\}) \\
[\![\textbf{true}]\!]_{assert}\sigma &= 1 \\
[\![\textbf{false}]\!]_{assert}\sigma &= 0 \\
[\![\neg p]\!]_{assert}\sigma &= \neg([\![p]\!]_{assert}\sigma) \\
[\![e \circledcirc e']\!]_{assert}\sigma &= [\![e]\!]_{intexp}\sigma \circledcirc [\![e']\!]_{intexp}\sigma \\
[\![p \oslash p']\!]_{assert}\sigma &= [\![p]\!]_{assert}\sigma \oslash [\![p']\!]_{assert}\sigma \\
[\![\forall v.p]\!]_{assert}\sigma &= \bigsqcap_{n\in\mathbb{Z}} [\![p]\!]_{assert}[\sigma \mid v : n] \\
[\![\exists v.p]\!]_{assert}\sigma &= \bigsqcup_{n\in\mathbb{Z}} [\![p]\!]_{assert}[\sigma \mid v : n]
\end{aligned}
$$

Ecuaciones semánticas

Observaciones

1. La notación $[\sigma \mid v : n]$ define la función

$$\sigma' \, w = \begin{cases} \sigma \, w & \text{si } w \neq v \\ n & \text{si } w = v \end{cases}$$

2. Podemos iterar la definición $[[\sigma \mid v : n] \mid w : m]$ que escribimos como $[\sigma \mid v : n \mid w : m]$.
3. A partir de ahora no pondremos los sub-índices en las ecuaciones semánticas (polimorfismo ad-hoc).
4. En la ecuación semántica de los cuantificadores queda claro que no importa el valor de $\sigma \, v$.

¿Por qué no importa? ¿Podemos decir algo más general?

Variables, libres y ligadas

Exploremos algunas frases

Consideremos las frases:

1. $\exists\ v, w\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$
2. $\exists\ u, x\ .\ (2 \leqslant u < z) \wedge (2 \leqslant x < z) \wedge u * x = z$
3. $\exists\ v\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$

Diremos que dos frases p y p' son ***equivalentes*** si su semántica es la misma, es decir si $[\![p]\!] = [\![p']\!]$.

¿Cuáles de esas tres frases son equivalentes entre sí?

Exploremos algunas frases

Las veamos en color:

1. $\exists\ v, w\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$
2. $\exists\ u, x\ .\ (2 \leqslant u < z) \wedge (2 \leqslant x < z) \wedge u * x = z$
3. $\exists\ v\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$

Observación
Una frase ***habla*** de sus variables libres: las dos primeras expresiones son predicados sobre z mientras que la segunda es sobre z y w.

Exploremos algunas frases

Definiciones

Ocurrencia Ligadora La ocurrencia de una variable se dice ***ligadora*** (binding) si está pegada a un cuantificador (binder); el ***alcance*** es el cuerpo de la cuantificación.

Ocurrencia Ligada Una ocurrencia de una variable v se dice ***ligada*** si está en el alcance de una ocurrencia ligadora de v.

Ocurrencia Libre Una ocurrencia de una variable v se dice ***libre*** si NO está en el alcance de una ocurrencia ligadora de v.

Variable Libre Una variable se dice ***libre*** si hay alguna ocurrencia libre de ella.

Expresión Cerrada Una expresión sin variables libres.

Exploremos algunas frases

Ejemplos
Repasemos esos conceptos con estos ejemplos.

1. $\exists\ v\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$
2. $\exists\ v\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge (\exists v\ .\ v > 5)$
3. $(\exists\ v\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z)) \wedge (v > 5)$

Exploremos algunas frases

Renombres

- Sea p la expresión
  $\exists\ v, w\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$
- Sea q la expresión
  $\exists\ u, x\ .\ (2 \leqslant u < z) \wedge (2 \leqslant x < z) \wedge u * x = z$

p y q son equivalentes porque una es un renombre de la otra: cambiamos la variable ligada v por la variable u y (al mismo tiempo) la variable w por x.

Exploremos algunas frases

Coincidencia (o no)

- Sea p la expresión
  $\exists\ v, w\ .\ (2 \leqslant v < z) \wedge (2 \leqslant w < z) \wedge v * w = z$

Ahora nos centramos en una única frase y recordemos la ecuación semántica para los cuantificadores. Supongamos que tenemos dos estados $\sigma, \sigma' \in \Sigma$, tales que $\sigma\ z = \sigma'\ z$.

¿Será que $[\![p]\!]\sigma = [\![p]\!]\sigma'$?

Variables Libres Formalmente

Para expresiones enteras

$$
\begin{aligned}
FV(\_) &: \langle \mathit{intexp} \rangle \to \mathcal{P}(\langle \mathit{var} \rangle) \\
FV(v) &= \{v\} \\
FV(\lfloor k \rfloor) &= \emptyset \\
FV(-e) &= FV(e) \\
FV(e \oplus e') &= FV(e) \cup FV(e')
\end{aligned}
$$

Variables Libres Formalmente

Para expresiones booleanas

$$
\begin{aligned}
FV(\_) &: \langle assert \rangle \to \mathcal{P}(\langle var \rangle) \\
FV(\textbf{true}) &= \emptyset \\
FV(\textbf{false}) &= \emptyset \\
FV(\neg p) &= FV(p) \\
FV(e \circledcirc e') &= FV(e) \cup FV(e') \\
FV(p \circledwedge p') &= FV(p) \cup FV(p') \\
FV(Qv.p) &= FV(p) \setminus \{v\}
\end{aligned}
$$

Sustitución

Sustitución

Como vimos en el ejemplo de los renombres, necesitamos una operación sobre las frases que nos permita cambiar variables por otras variables y, más en general, por expresiones enteras.

Una ***sustitución*** es un mapeo de variables a expresiones enteras. Usaremos $\Delta = \langle var \rangle \to \langle intexp \rangle$. Una sustitución es similar a un estado, en tanto que asigna posibles "valores" a las variables.

Podemos definir un operador de sustitución que deja intacta la estructura de la frase ***instanciando*** las variables libres con lo que indica la sustitución.

Sustitución

Sin embargo, hay que tener cuidado para no cambiar el sentido de la frase: Sabemos que

$$\forall \mathsf{x}.(\exists \mathsf{y}\ .\mathsf{y} > \mathsf{x})$$

es cierto en $\mathbb{Z}$; entonces podríamos concluir que es válido para un valor concreto de x, por ejemplo mapeando $x \mapsto y + 1$ lo que daría lugar a:

$$\exists \mathsf{y}\ .\mathsf{y} > \mathsf{y} + 1$$

El problema es que al hacer el reemplazo sintáctico estamos capturando a y.

Sustitución

Definición
Sea $\delta \in \Delta = \langle var \rangle \to \langle intexp \rangle$.

$$
\begin{aligned}
\_/\_ &: \theta \times \Delta \to \theta \\
v/\delta &= \delta\, v \\
\lfloor k \rfloor/\delta &= \lfloor k \rfloor \\
(\neg p)/\delta &= \neg(p/\delta) \\
(p \circledcirc p')/\delta &= p/\delta \circledcirc p'/\delta \\
(Qv.p)/\delta &= Qv_{new}.p/[\delta \mid v : v_{new}] \\
&\quad \text{donde } v_{new} \notin \bigcup_{w \in FV(p)\setminus\{v\}} FV(\delta\, w)
\end{aligned}
$$

Sustitución

Observaciones

1. La definición evita la captura pidiendo que v_{new} no vaya a capturar ninguna de las variables que introducirá la sustitución en el cuerpo del cuantificador.
2. ¿Quién es v_{new}? Si v no está en el conjunto de nuevas variables libres, entonces es v. Si no, será la primera variable que no ocurra en ese conjunto.
3. Es decir, asumimos que hay algún orden (lo cual es gratis porque asumíamos que $\langle var \rangle$ era numerable).
4. En el Reynolds aparece c_{var} que es la función que inyecta variables como expresiones; es una especie de identidad de las sustituciones.

Teoremas

Coincidencia

Si dos estados σ y σ' coinciden en las variables libres de p, entonces $[\![p]\!]\sigma = [\![p]\!]\sigma'$.

Teorema
Para todo tipo θ, para toda frase p de tipo θ. Si para toda $v \in FV(p)$, $\sigma\, v = \sigma'\, v$; entonces $[\![p]\!]\sigma = [\![p]\!]\sigma'$.

Demostración.
Primero lo probamos para $\langle \mathit{intexp} \rangle$ y luego para $\langle \mathit{assert} \rangle$. En cada caso por inducción en p. □

Sustitución

Si un estado σ' hace el trabajo combinado de un estado σ y una sustición δ, restringidas a las variables libres de p; entonces $[\![p/\delta]\!]\sigma = [\![p]\!]\sigma'$.

Teorema
Para todo tipo θ, para toda frase p de tipo θ. Para todas $\sigma, \sigma' \in \Sigma$ y $\delta \in \langle var \rangle \to \langle intexp \rangle$. Si para toda $v \in FV(p)$, $[\![\delta\, v]\!]\sigma = \sigma'\, v$; entonces $[\![p/\delta]\!]\sigma = [\![p]\!]\sigma'$.

Demostración.
Primero lo probamos para $\langle intexp \rangle$ y luego para $\langle assert \rangle$. En cada caso por inducción en p. □

Sustitución finita

Sustituir algunas variables por expresiones y calcular la semántica es lo mismo que calcular la semántica en el estado donde esas variables vales la semántica de las expresiones.

Teorema

$$[\![p/[v_0 : e_0 \mid \ldots \mid v_{n-1} : e_{n-1}]]\!]\sigma = [\![p]\!][\sigma \mid v_0 : [\![e_0]\!]\sigma \mid \ldots \mid v_{n-1} : [\![e_{n-1}]\!]\sigma]$$

Demostración.
Sale aplicando el teorema de sustitución. □

Renombre

Si hago un renombre de una variable ligada, el significado no cambia.

Teorema
$[\![\forall u.p/[v : u]]\!] = [\![\forall v.p]\!]$.

Demostración.
Sale aplicando el teorema de sustitución. □