MFI 2001/2002 - Compitino n. 1

Si consideri il linguaggio giocattolo orientato alla definizione di circuiti dato dalla seguente grammatica $\mathbf{G}$:

$$
\begin{array}{lcl}
\texttt{Prog} & ::= & \textbf{input}\ \texttt{Ids};\ \textbf{output}\ \texttt{Ids};\ \textbf{use gates}\ \texttt{Decs}\ \{\texttt{Links}\}. \\
\texttt{Ids} & ::= & \texttt{Id} \mid \texttt{Id}\ \texttt{Ids}. \\
\texttt{Id} & ::= & \ldots. \\
\texttt{Decs} & ::= & \texttt{Dec} \mid \texttt{Dec}\ \texttt{Decs}. \\
\texttt{Dec} & ::= & \texttt{Type}\ \texttt{Id};. \\
\texttt{Type} & ::= & \textbf{and_gate} \mid \textbf{or_gate} \mid \textbf{not_gate}. \\
\texttt{Links} & ::= & \texttt{Link} \mid \texttt{Link}\ \texttt{Links}. \\
\texttt{Link} & ::= & \texttt{Exp} \rightarrow \texttt{Exp};. \\
\texttt{Exp} & ::= & \texttt{Id} \mid \mathbf{in}_1(\texttt{Id}) \mid \mathbf{in}_2(\texttt{Id}) \mid \mathbf{out}(\texttt{Id}).
\end{array}
$$

L'idea intuitiva è che un programma in questo linguaggio dichiara assegna dei nomi logici ad input, output e singoli gate del circuito e definisce, mediante i link, le connessioni fra gli input e gli output di ciascun gate.

Pertanto, oltre ai vincoli statici "standard", quali divieto di doppie dichiarazioni, si avranno dei vincoli statici che esprimano l'idea un link rappresenta la connessione fisica che trasferisce il valore prodotto dalla prima espressione alla seconda. Quindi un link dovrà associare qualcosa che "produce" un valore (sia esso il canale di input del circuito o l'output di un gate) a qualcosa che può "usare" (ovvero uno degli output del circuito o un ingresso di un gate). Inoltre, bisognerà verificare che non ci siano collisioni (due "fili" che danno input nello stesso punto) né interruzioni nel circuito (un input di un gate che non è connesso agli input del circuito).

Si dia la semantica statica di questo linguaggio, usando l'insieme ausiliario $\texttt{Env}^S = [L_{\texttt{Id}}(\mathbf{G}) \rightarrow_P \texttt{Types}]_{Fin}$, dove $\texttt{Types} = \{\texttt{T}_{\texttt{in}}, \texttt{T}_{\texttt{out}}, \texttt{T}_{\texttt{and}}, \texttt{T}_{\texttt{or}}, \texttt{T}_{\texttt{not}}\}$. Ulteriori vincoli statici sono dettagliati nel seguito.

1. (punti 3)

   $[\![_]\!]^S_{\texttt{Ids}} : L_{\texttt{Ids}}(\mathbf{G}) \times \texttt{Env}^S \times \{\texttt{T}_{\texttt{in}}, \texttt{T}_{\texttt{out}}\} \rightarrow_P \texttt{Env}^S$ verifica che ciascun identificatore sia nuovo e lo aggiunge all'ambiente con il tipo indicato (ultimo argomento);

2. (punti 3)

   $[\![_]\!]^S_{\texttt{Decs}} : L_{\texttt{Decs}}(\mathbf{G}) \times \texttt{Env}^S \rightarrow_P \texttt{Env}^S$ verifica che ciascun identificatore sia nuovo e lo aggiunge all'ambiente con il tipo con cui è definito;

3. (punti 6)

   $[\![_]\!]^S_{\texttt{Exp}} : L_{\texttt{Exp}}(\mathbf{G}) \times \texttt{Env}^S \to_P \{\texttt{T}_\texttt{d}, \texttt{T}_\texttt{s}\}$ associa ad ogni espressione corretta il il suo tipo. Per essere corretta un'espressione deve soddisfare i seguenti vincoli:

   - ciascun identificatore deve essere noto nell'ambiente ed usato in maniera consistente col tipo associatogli dall'ambiente;
   - $\mathbf{in}_1$(g) e **out**(g) sono corrette solo se g è un gate;
   - $\mathbf{in}_2$(g) è corretta solo se g è un gate binario (cioè un **and_gate** o un **or_gate**);

   Un'espressione è di tipo $\texttt{T}_\texttt{d}$ se è della forma accettabile come parte destra di un link, oppure $\texttt{T}_\texttt{s}$ se è della forma accettabile come parte sinistra di un link, come definito dai seguenti vincoli:

   - un'espressione è accettabile come parte parte sinistra di un link solo se è uno degli input del programma, oppure è della forma **out**(g);
   - un'espressione è accettabile come parte parte destra di un link solo se è uno degli output del programma, oppure un'espressione (corretta) della forma $\mathbf{in}_1$(g) o $\mathbf{in}_2$(g);

4. (punti 8)

   $[\![_]\!]^S_{\texttt{Links}} : L_{\texttt{Links}}(\mathbf{G}) \times \texttt{Env}^S \times \wp(L_{\texttt{Exp}}(\mathbf{G})) \to_P \wp(L_{\texttt{Exp}}(\mathbf{G}))$ dove l'ultimo parametro rappresenta le espressioni già *saturate*, cioè già usate come parti destre in links precedenti e il risultato, se corretto, sarà l'insieme delle espressioni saturate finora.

   Vincoli su ciascun link $\texttt{l} = e_1 \to e_2$;

   - e_1 deve essere accettabile come parte parte sinistra di un link e deve essere saturata;
   - e_2 deve essere accettabile come parte parte destra di un link e NON deve saturata.

5. (punti 4)

   $[\![_]\!]^S_{\texttt{Prog}} : L_{\texttt{Prog}}(\mathbf{G}) \to_P \{*\}$ oltre a verificare che tutte le parti siano corrette e coerenti, deve anche controllare che tutti gli output del circuito complessivo siano saturati.