Linguaggi e compilatori, primo progetto

Sistemi di riscrittura

Si stabilisca (argomentando opportunamente) se il seguente TRS è lineare destro/sinistro, duplicante, eliminante, collassante; se è constructor-based (definendo un opportuno insieme di costruttori) e se è ortogonale.

$$
\begin{aligned}
f(h(2,x),y) &\to j(y,2)\\
f(x,j(2,y) &\to x\\
g(h(4,x),y) &\to h(y,j(2,y))\\
g(f(x,y),x) &\to f(y,x)
\end{aligned}
$$

Parser top-dowun

Data la grammatica

$$
\begin{aligned}
S &\to B \mid S \texttt{ and } S \mid \texttt{not } S \mid \texttt{(} S \texttt{)}\\
B &\to N \texttt{ < } N\\
N &\to \texttt{id} \mid - N \mid N \cdot N
\end{aligned}
$$

1. Definire una grammatica L non ambigua, equivalente alla grammatica data e in cui gli operatori binari $\cdot$, *and* siano associativi a destra e in cui *not* abbia priorità su *and* e $\cdot$ abbia priorità su $-$.
2. Definire una grammatica LL(1) equivalente alla grammatica L.
3. Costruirne il parser top-down.
4. Mostrare l'esecuzione del parser sull'input:

   $$(\texttt{id} < \texttt{or } - \texttt{id})\ \texttt{not and id} < \texttt{id} \cdot \texttt{id and id id})$$

   utilizzando opportuni passi di error recovery.

Parser bottom-up

Data la grammatica

$$
\begin{aligned}
S &\to \texttt{begin } SL \texttt{ end} \mid \texttt{id = num} \mid \texttt{goto id} \mid \texttt{id :} \ S\\
SL &\to S \mid SL \texttt{ ; } S
\end{aligned}
$$

1. Costruire i parsers SLR e LALR.
2. Mostrare l'esecuzione di suddetti parsers sull'input

```
begin
 c=
goto b;
b=5;
end
```

utilizzando opportuni passi di error recovery. Qualora ci fossero conflitti nelle rispettive tabelle, si utilizzi la convenzione di prediligere lo shift sul reduce e di scegliere il reduce con regola testualmente precedente.