Tutorato AFL

Linpeng Zhang

3 giugno 2019

Sommario

Per errori/dubbi/problemi: linpeng.zhang@studenti.unipd.it.

Indice

1 Lez12

Riscrivere le soluzioni dei professori sarebbe uno spreco di tempo, motivo per cui ho preferito fare un "riassunto informale".

1.1 Riassunto informale

1.1.1 Automi a pila

1. un automa a pila nondeterministico con qualsiasi tipo di accettazione riconosce tutte i CFL;
2. un automa a pila deterministico DPDA ha:
   (a) al più uno stato per ogni tripla (stato, input, cima della pila):
   (b) non può decidere se consumare o meno l'input. Cioè se l'automa ha una transizione per (q,ϵ,X) allora non può averne una per (q, unastringa, X).
3. un DPDA per stato finale riconosce tutti i linguaggi regolari, ma non tutti i CFL (ad esempio non L_pal);

4. un DPDA per stack vuoto riconosce un sottoinsieme dei CFL che hanno la proprietà del prefisso: ovvero un prefisso di una parola del linguaggio non può a sua volta essere una parola del linguaggio;
5. sia P un DPDA. Allora L(P) non è ambiguo.

1.1.2 Indecidibilità

1. ci sono linguaggi non RE, detti anche indecibili. Lo sono ad esempio $L_d, comp(L_u), L_e$;
2. ci sono linguaggi RE: data una TM che riconosce un linguaggio RE, se riceve in input una stringa nel linguaggio la TM la accetta e termina; se riceve in input una stringa non nel linguaggio, la TM non necessariamente termina. L_u, L_{ne} lo sono.
3. ci sono linguaggi RE ricorsivi: data una TM che riconosce un linguaggio RE ricorsivi, se riceve in input una stringa nel linguaggio la TM la accetta e termina; se riceve in input una stringa non nel linguaggio, la TM certamente termina;

1.1.3 Trattabilità

1. abbiamo trattato solamente i problemi di decisione, ma ciò non è limitativo in quanto molti problemi possono ricondursi ad una variante "decisionale";
2. un problema è P se una macchina di Turing deterministica lo risolve in tempo polinomiale;
3. un problema è NP se una macchina di Turing nondeterministica lo risolve in tempo polinomiale, oppure se esiste un certificato della soluzione che può essere verificato in tempo polinomiale;
4. un problema è NP-hard se può essere utilizzato come sottoprocedura per risolvere qualsiasi problema in NP con una eventuale trasformazione in tempo polinomiale;
5. un problema è NP-completo se è NP e NP-hard. Quindi per dimostrare che un P' è NP-completo dovrete:
   (a) dire com'è fatto un certificato e come verificarlo in tempo polinomiale;
   (b) fare una riduzione di un problema NP-completo o NP-hard a P'. In particolare la trasformazione h(x) dell'input deve avvenire in tempo polinomiale e dovete dimostrare come risolvere il problema NP-completo sull'input originario sia equivalente a risolvere il problema P' sull'input trasformato.