Capitolo 1

Prerequisiti

In questo capitolo verranno descritte le nozioni di base necessarie per comprendere il lavoro svolto. In particolare, verranno introdotti i concetti di logica proposizionale e del primo ordine, definita come estensione della prima. Nell'ultimo paragrafo del capitolo verrà descritto in che modo le formule di logica del primo ordine possono essere rappresentate in un formato di file, per poi essere processate come input da un theorem prover. Lo scopo di questo capitolo è quello di accennare la teoria logica utilizzata nell'implementazione di vampire e della procedura di decisione per i Binding-Fragments. Perciò, verranno date per scontate nozioni di teoria degli insiemi, algebra booleana e teoria dei linguaggi formali.

1.1 Logica Proposizionale

1.1.1 Formule

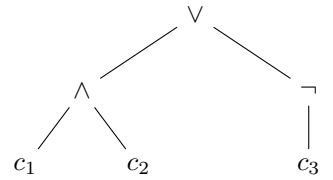
Sia $\Sigma_c = \{c_1, c_2, ...\}$ un insieme di simboli di costante, $\Sigma = \{\wedge, \vee, \neg, (,), \top, \bot\} \cup \Sigma_c$ è detto alfabeto della logica proposizionale. Con queste premesse possiamo definire come formule della logica proposizionale il linguaggio generato dalla grammatica Context Free seguente:

$$\varphi := \top \mid \bot \mid C \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi)$$

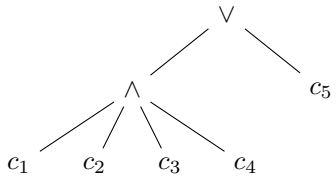
Dove $C \in \Sigma_c$ è un simbolo di costante. Con la funzione $const(\gamma) \to \Sigma_c$ si indica la funzione che associa a ogni formula γ l'insieme dei suoi simboli di costante. Vengono inoltre introdotti i seguenti simboli come abbreviazioni:

- $(\gamma \Rightarrow \kappa)$ per $(\neg\gamma \vee \kappa)$
- $(\gamma \Leftrightarrow \kappa)$ per $((\gamma \Rightarrow \kappa) \wedge (\kappa \Rightarrow \gamma))$
- $(\gamma \oplus \kappa)$ per $\neg(\gamma \Leftrightarrow \kappa)$

È possibile rappresentare una qualunque formula attraverso il proprio albero di derivazione. Questo albero verrà chiamato in seguito anche *albero sintattico* della formula. Ad esempio, la formula $(c_1 \wedge c_2) \vee \neg c_3$ può essere rappresentata dal seguente albero sintattico:



La radice dell'albero è detta *connettivo principale*. Per compattezza, grazie alla proprietà associativa di $\wedge$ e $\vee$, è possibile omettere le parentesi, es. $(c_1 \wedge (c_2 \wedge (c_3 \wedge c_4))) \vee c_5$ può essere scritto come $(c_1 \wedge c_2 \wedge c_3 \wedge c_4) \vee c_5$. Allo stesso modo, nell'albero sintattico della formula è possibile compattare le catene di $\wedge$ e $\vee$ come figli di un unico nodo:



Questa è una caratteristica molto importante, in quanto non solo permette di risparmiare inchiostro, ma consente di vedere $\wedge$ e $\vee$ non più come operatori binari ma come operatori n-ari. A livello implementativo, ciò si traduce in un minor impatto in memoria, visite all'albero più veloci e algoritmi di manipolazione più semplici. Si consideri ad esempio di voler ricercare la foglia più a sinistra nell'albero di derivazione della seguente formula $((...(((c_1 \wedge c_2) \wedge c_3) \wedge c_4) \wedge ...) \wedge c_n)$. Senza compattazione, l'algoritmo di ricerca impiegherebbe $O(n)$ operazioni, mentre con la compattazione $O(1)$.

1.1.2 Assegnamenti

1.1.3 Forme Normali

1.1.4 Naming

1.2 Logica del primo ordine

1.2.1 Termini

1.2.2 Formule

1.2.3 Semantica

1.2.4 Forme Normali

1.2.5 Skolemizzazione

1.2.6 Unificazione

1.3 Soddisfacibilità e Validità

1.4 Resolution

1.5 Il formato TPTP