Capitolo 1

Algoritmo di decisione di Frammenti Binding

Nella sezione **??**, sono stati esaminati i teoremi di Gödel e Church, mentre nella sezione **??** sono state viste alcune delle loro conseguenze. La logica del primo ordine è intrinsecamente indecidibile; tuttavia, è possibile identificare alcune sue componenti che risultano decidibili. Queste componenti sono dette *Frammenti* della logica del primo ordine. Si pensi ad esempio ai risultati di Herbrand citati nella sezione **??**. Se una formula non contiene funzioni ed è universalmente quantificata allora l'universo di Herbrand è finito e vi sono un numero finito di possibili instanziazioni ground. In questo caso determinare la soddisfacibilità di una formula di questo tipo è riducibie al problema della soddisfacibilità proposizionale che è notoriamente decidibile. In letteratura questo frammento è noto come *Bernays–Schönfinkel Fragment*. Altre esempi di frammenti decidibili sono il *Monadic Fragment*, il *Two-variable Fragment*, *Unary negation fragment* e il *Guarded Fragment*. In questo capitolo verrà descritta una famiglia di frammenti relativamente recente chiamata *Binding Fragments* [**?**] [**?**].

1.1 Tassonomia dei Frammenti Binding

Si dice che una formula del primo ordine appartiene alla classe *Boolean Binding* (BB) se generata dalla seguente grammatica:

$$\varphi := \top \mid \bot \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \mathcal{P}(\psi)$$
$$\psi := \rho \mid (\psi \vee \psi) \mid (\psi \wedge \psi)$$

Dove $\mathcal{P}$ è un prefisso di quantificatori e ρ è una combinazione booleana di letterali che hanno come argomento tutti la stessa lista di termini. Una formula di questo tipo verrà chiamata con il nome τ-Binding, dove τ indica la lista di termini comune. Ad esempio sono $(f_1(x_1), f_2)$-Binding le formule: $p_1(f_1(x_1), f_2)$, $p_1(f_1(x_1), f_2) \vee \neg p_3(f_1(x_1), f_2)$. Per semplicità di scrittura è possibile omettere la lista di termini comune e posizionarla in notazione postfissa:

$$p_1(f_1(x_1), f_2) \vee \neg p_3(f_1(x_1), f_2) \text{ diventa } (p_1 \vee \neg p_3)(f_1(x_1), f_2)$$

Con $\mathcal{B}^\tau$ verrà indicato l'insieme di tutte le formule τ-Binding. Si definisce la funzione $term : \mathcal{B}^\tau \to T^n$ che associa ogni τ-Binding alla sua lista di termini comune τ. Ad esempio $term((p_1 \vee \neg p_3)(f_1(x_1), f_2)) = (f_1(x_1), f_2)$. Verranno chiamati impropriamente τ-Binding anche formule universalmente quantificate la cui matrice è un τ-Binding. In questo caso ci si riferisce esclusivamente alla matrice della formula eliminando i quantificatori.

I frammenti Binding possono essere ottenuti restringendo le regole di ψ:

- Il frammento *One Binding* (1B) viene ottenuto restringendo la seconda formula a $\psi := \rho$
- Il frammento *Conjunctive Binding* (CB o $\wedge$B) viene ottenuto restringendo la seconda formula a $\psi := \rho \mid (\psi \wedge \psi)$
- Il frammento *Disjunctive Binding* (DB o $\vee$B) viene ottenuto restringendo la seconda formula a $\psi := \rho \mid (\psi \vee \psi)$

Un'istanza particolare del frammento 1B è quando la formula non contiene quantificatori esistenziali. Una formula 1B con soli prefissi universali viene detta del frammento *Universal One Binding* ($\forall 1B$).

1.2 Soddisfacibilità dei frammenti Binding

In questa sezione verrà analizzato il problema della soddisfacibilità dei frammenti binding. In particolare verrà descritto l'algoritmo di decisione per i frammenti 1B e CB che è il soggetto principale dello studio di questa tesi.

Data una formula del frammento 1B è facile osservare che il processo di skolemizzazione converte la formula in formato $\forall 1B$. Se si applica la stessa procedura ad una formula CB, le sottoformule generate dalla regola ψ saranno del tipo: $\mathcal{P}(\rho_1 \wedge ... \wedge \rho_n)$ con $\mathcal{P}$ un prefisso universale e $(\rho_1 \wedge ... \wedge \rho_n)$ τ-Binding. In questo caso è possibile distribuire il '$\forall$' sui vari τ-Binding e si ottiene così una formula equisoddisfacibile in formato $\forall$1B.

Teorema: Decidibilità dei frammenti 1B e CB 1.2.1. *I frammenti 1B e CB sono frammenti decidibili del primo ordine.*

Una dimostrazione dettagliata di questo teorema può essere trovata nell'articolo [TODO da citare]. Si può osservare che il processo di clausificazione del primo ordine porta alla generazione di una formula equisoddisfacibile che rispetta il formato DB. Ne consegue immediatamente per il teorema di Church:

Teorema: Indecidibilità del frammento Disjunctive Binding 1.2.2. *Il frammento DB è un frammento indecidibile del primo ordine.*

Dimostrazione. Per assurdo Esiste un algoritmo di decisione totale S per formule del frammento DB. Data una qualunque formula φ è possibile trasformarla in una equisoddisfacibile in formato CNF. Se si distribuisce il quantificatore universale sulle clausole si ottiene una formula φ' che rispetta i requisiti sintattici del frammento DB. S è quindi una procedura di decisione totale per tutta la logica del primo ordine ma ciò è in contraddizione con il teorema di Church. □

Il processo di skolemizzazione consente di concentrarsi sullo studio del frammento $\forall$1B per la risoluzione del problema della soddisfacibilità. Prima di descrivere l'algoritmo bisogna introdurre tre nuovi concetti: L'Unificazione per τ-Binding, Implicante di una formula del primo ordine e la conversione booleana di un τ-Binding. Data una formula del primo ordine φ per Implicante di φ si intende la conversione del primo ordine di un implicante della 'struttura proposizionale esterna'. ad esempio la formula $\forall x_1(p_1(x_1) \vee p_2(x_1)) \wedge (p_1(f_1) \vee \exists x_2(p_3(x_2))) \wedge \neg p_1(f_1) \wedge \exists x_2(p_3(x_2))$ ha la seguente struttura booleana $s_1 \wedge (s_2 \vee s_3) \wedge \neg s_2 \wedge s_3$. Un implicante (e anche il solo) di questa formula è l'insieme $\{s_1, s_3\}$ che ri-convertito nel primo ordine diventa l'insieme $\{\forall x_1(p_1(x_1) \vee p_2(x_1)), \exists x_2(p_3(x_2))\}$. In questo caso è stata creata implicitamente un bi-mappa tra costanti proposizionali e formule del primo ordine:

- $s_1 \rightleftarrows \forall x_1(p_1(x_1) \vee p_2(x_1))$
- $s_2 \rightleftarrows p_1(f_1)$

- $s_3 \rightleftarrows \exists x_2(p_3(x_2))$

Un τ_1-Biding e un τ_2-Biding sono detti unificabili se e solo se l'insieme congiunto di tutti i loro letterali è unificabile. Si può anche dire che sono unificabili sse le due liste τ_1 e τ_2 hanno la stessa lunghezza n e dato un qualunque predicato p n-ario $p(\tau_1)$ e $p(\tau_2)$ sono unificabili. Una insieme di τ-Biding è unificabile sse esiste una sostituzione che unifica a due a due tutti gli elementi dell'insieme. Dato un τ-Binding ϕ la sua conversione booleana $bool(\phi)$ è una formula proposizionale che si ottiene da ϕ mantenendo la sua struttura proposizionale, eliminando gli argomenti dai letterali e convertendo i simboli di predicato in simboli di costante con lo stesso indice. Ad esempio il τ-Binding $((p_1 \wedge p_4) \vee p_2 \vee \neg p_4)(\tau)$ viene convertito nella seguente formula proposizionale $(s_1 \wedge s_4) \vee s_2 \vee \neg s_4$

A questo punto è possibile enunciare il teorema di caratterizzazione della soddisfacibilità del frammento $\forall$1B.

Teorema: Caratterizzazione della soddisfacibilità per il frammento $\forall$1B 1.2.3. *Data una formula φ del frammento $\forall$1B, φ è soddisfacibile se e solo se:*
Esiste un implicante I dove: per ogni sottoinsieme $U \subseteq I$ di τ-Binding, se $U = \{\gamma_1, ..., \gamma_n\}$ è unificabile allora la formula proposizionale $bool(\gamma_1) \wedge ... \wedge bool(\gamma_n)$ è soddisfacibile.

Dal teorema appena descritto si estrapola intuitivamente l'algoritmo per la soddisfacibilità delle formule del frammento:

Algorithm 1: Algoritmo per la soddisfacibilità del frammento $\forall$1B

Firma: oneBindingAlgorithm(φ)
Input: φ una formula $\forall$1B
Output: $\top$ o $\bot$

```
foreach I Implicant of φ do
    res := ⊤;
    foreach (U := {γ1, ..., γn}) ⊆ I do
        if U is unifiable then
            if bool(γ1) ∧ ... ∧ bool(γn) is not satisfiable then
                res := ⊥;
                Break;
            end
        end
    end
    if res = ⊤ then
        return ⊤
    end
end
return ⊥
```

I prossimi capitoli si concentreranno sullo studio dei dettagli tecnici per l'implementazione di questo algoritmo, con annesse osservazioni sulle sfide implementative e una analisi dei risultati sperimentali ottenuti.