Proof Search in Propositional Linear Logic via Boolean Constraints Satisfaction

Laurea Triennale in Informatica

Martino D'Adda (964827)

16 Luglio 2024

UNIVERSITÀ DEGLI STUDI DI MILANO

Indice

1 Introduzione

- Introduzione
- Il calcolo
- Conclusione

Calcolo dei sequenti

1 Introduzione

Il calcolo dei sequenti (G. Gentzen, 1934) è un formalismo per rappresentare un calcolo logico e strutturarne dimostrazioni. I concetti principali sono:

Calcolo dei sequenti

1 Introduzione

Il calcolo dei sequenti (G. Gentzen, 1934) è un formalismo per rappresentare un calcolo logico e strutturarne dimostrazioni. I concetti principali sono:

- il sequente $\Delta \vdash \Gamma$, di solito interpretato come

$$\delta_1 \wedge \cdots \wedge \delta_n \to \gamma_1 \vee \cdots \vee \gamma_m$$

Calcolo dei sequenti

1 Introduzione

Il calcolo dei sequenti (G. Gentzen, 1934) è un formalismo per rappresentare un calcolo logico e strutturarne dimostrazioni. I concetti principali sono:

- il sequente $\Delta \vdash \Gamma$, di solito interpretato come

$$\delta_1 \wedge \cdots \wedge \delta_n \to \gamma_1 \vee \cdots \vee \gamma_m$$

- la regola, per manipolare i sequenti
  - descrive la semantica di un connettivo (introduzione e eliminazione):

$$\text{intro}\wedge \frac{\Delta \vdash \phi', \Gamma \qquad \Delta \vdash \phi'', \Gamma}{\Delta \vdash \phi' \wedge \phi'', \Gamma} \qquad \text{elim}\wedge \frac{\Delta, \phi', \phi'' \vdash \Gamma}{\Delta, \phi' \wedge \phi'' \vdash \Gamma}$$

  - descrive come trattare le formule all'interno del sequente (regole strutturali).

Calcolo dei sequenti

1 Introduzione

Il calcolo dei sequenti (G. Gentzen, 1934) è un formalismo per rappresentare un calcolo logico e strutturarne dimostrazioni. I concetti principali sono:

- il sequente $\Delta \vdash \Gamma$, di solito interpretato come

$$\delta_1 \wedge \cdots \wedge \delta_n \to \gamma_1 \vee \cdots \vee \gamma_m$$

- la regola, per manipolare i sequenti
  - descrive la semantica di un connettivo (introduzione e eliminazione):

$$\text{intro}\wedge \frac{\Delta \vdash \phi', \Gamma \qquad \Delta \vdash \phi'', \Gamma}{\Delta \vdash \phi' \wedge \phi'', \Gamma} \qquad \text{elim}\wedge \frac{\Delta, \phi', \phi'' \vdash \Gamma}{\Delta, \phi' \wedge \phi'' \vdash \Gamma}$$

  - descrive come trattare le formule all'interno del sequente (regole strutturali).
- la dimostrazione, un albero avente come radice il sequente da dimostrare e ottenuto concatenando regole.

Theorem prover

1 Introduzione

Un **theorem prover** è un programma che, dato un sequente, prova a costruirne la dimostrazione. Nello specifico la versione bottom-up:

1. inizia con un singolo *goal*, cioè il sequente iniziale
2. sceglie un nuovo goal;
3. lo scompone applicando una regola “al contrario” ad una delle formule nel goal:
   - non-determinismo don’t-care: tutte le scelte sono equivalenti equivalenti;
   - non-determinismo don’t-know: non tutte le scelte sono equivalenti e può essere necessario del backtracking;
4. aggiunge i nuovi goal al working-set;
5. si ripete dal passo 2 fino a che ci sono goal.

Focusing

1 Introduzione

Le regole di un calcolo possono essere suddivise in due classi:

- invertibili: l'ordine di applicazione non è importante;
- non invertibili: sono necessarie scelte che potrebbero necessitare backtracking.

Il **focusing** è una tecnica che suddivide la dimostrazione in due fasi che si alternano:

- una fase (asincrona) in cui si applicano eagerly solo regole invertibili;
- una fase (sincrona) in cui ci si concentra su una formula e si continuano ad applicare regole non invertibili.

In questo modo si minimizza il non-determinismo don't-care.

Logica lineare

1 Introduzione

La **logica lineare** nasce dall'analisi delle regole strutturali e dall'abbandono di due di esse:

- *weakening*: nel sequente si possono sempre aggiungere formule;
- *contraction*: due copie della stessa formula possono essere contratte.

Nella logica che ne risulta:

- è generalmente proibita la duplicazione o l'eliminazione di formule;
- le formule possono essere viste come **risorse**;
- ogni connettivo ammette due versioni, una additiva ed una moltiplicativa, corrispondenti a due diverse interpretazioni dei connettivi classici;

Degli speciali connettivi ($?, !$), chiamati esponenziali, permettono di localizzare la copia e l'eliminazione di formule.

Gestione delle risorse

Durante il proof searching bottom-up in logica lineare un'ulteriore fonte di non-determinismo è la gestione delle risorse.

$$\dfrac{\dfrac{A\ \dfrac{}{\beta \vdash \beta} \qquad A\ \dfrac{}{\alpha \vdash \alpha}}{\alpha, \beta \vdash \beta \otimes \alpha}\otimes_R}{\alpha \otimes \beta \vdash \beta \otimes \alpha}\otimes_L$$

Nello specifico la scomposizione dei moltiplicativi spesso comporta un'operazione chiamata **splitting**, cioè il partizionamento del contesto.

Indice

2 Il calcolo

- Introduzione
- Il calcolo
- Conclusione

Calcolo dei vincoli

2 Il calcolo

Nel 2001 D.Pym e J.Harland propongono un di gestione delle risorse basato su vincoli booleani. Questi

- sono generati durante la proof-search a partire da espressioni associate alle formule;
- possono essere solo di due tipi:
  - una certa formula è stata usata in questo goal, e non può essere usata altrove;
  - una certa formula non è stata usata in questo goal, e deve essere usata altrove;

In questo modo è possibile non partizionare fisicamente il contesto

$$\dfrac{\dfrac{A\ \dfrac{}{\alpha, \beta \vdash \beta}\quad A\ \dfrac{}{\alpha, \beta \vdash \alpha}}{\otimes_R\ \alpha, \beta \vdash \beta \otimes \alpha}}{\otimes_L\ \alpha \otimes \beta \vdash \beta \otimes \alpha}$$

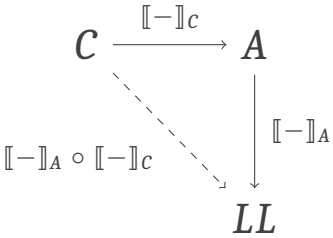
Calcolo dei vincoli cont'd

2 Il calcolo

Rendiamo il calcolo dei vincoli focused, e di questo nuovo calcolo mostriamo la correttezza esibendo una traduzione tale che il diagramma a lato commuta.
Con:

- C il nostro calcolo;
- A il calcolo focused senza constraint;
- LL il calcolo classico della logica lineare.

e $[\![-]\!]$ le rispettive traduzioni.



Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

- un'implementazione in SWI-Prolog;

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

- un'implementazione in SWI-Prolog;
- un generatore di test in OCaml;

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

- un'implementazione in SWI-Prolog;
- un generatore di test in OCaml;
- una libreria per il testing e il benchmarking in Python.

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

- un'implementazione in SWI-Prolog;
- un generatore di test in OCaml;
- una libreria per il testing e il benchmarking in Python.

Infine l'infrastruttura è stata gestita con Nix.

SWI-Prolog

2 Il calcolo

Nello specifico è stato scelto SWI-Prolog per:

- la sua capacità di esprimere le regole in modo dichiarativo;
- il suo supporto di prim'ordine per il backtracking;
- le sue librerie per il constraint logic programming (CLP), che offrono una elegante interfaccia verso risolutori di vincoli booleani;
- l'unificazione permette di gestire in automatico la propagazione degli assegnamenti delle variabili booleane.

Indice

3 Conclusione

- Introduzione
- Il calcolo
- Conclusione

Risultati

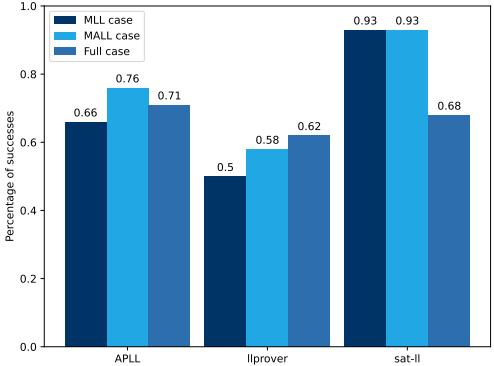
3 Conclusione

Il prover è stato confrontato con altri due prover:

- llprover (Prolog, 1997)
- APLL (OCaml, 2019)

Le fonti dei test sono state principalmente due:

- test generati (MLL e MALL);
- un dataset di traduzioni dalla logica intuizionista per i test esponenziali.



Conclusione e lavori futuri

3 Conclusione

Quindi del calcolo dei vincoli

- abbiamo fornito un nuovo calcolo che utilizzi il focusing, di cui abbiamo dimostrato la correttezza;
- abbiamo fornito un'implementazione in Prolog;
- abbiamo confrontato questa implementazione con altri prover simili e mostrato che nel caso moltiplicativo si ottengono buoni risultati.

Possibili sviluppi futuri possono essere:

- affinamento dell'algoritmo nel caso esponenziale;
- scrittura di altri calcoli e prover simili per altre logiche con moltiplicativi (ILL, BI, ...).

Fine.