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

Calcolo dei sequenti

1 Introduzione

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

Calcolo dei sequenti

1 Introduzione

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

- il sequente $\Delta \vdash \Gamma$, che rappresenta una implicazione tra la congiunzione e la disgiunzione di sequenze di formule;

Calcolo dei sequenti

1 Introduzione

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

- il sequente $\Delta \vdash \Gamma$, che rappresenta una implicazione tra la congiunzione e la disgiunzione di sequenze di formule;
- la regola, che descrive come si possono manipolare i sequenti, ad esempio

$$\wedge \frac{\overbrace{\Delta \vdash \phi', \Gamma}^{\Pi'} \qquad \overbrace{\Delta \vdash \phi'', \Gamma}^{\Pi''}}{\underbrace{\Delta \vdash \phi' \wedge \phi'', \Gamma}_{\Pi}}$$

  stabilisce che se valgono Π' e Π'', allora vale Π.

Calcolo dei sequenti cont'd

Una dimostrazione consiste in un albero avente come radice il sequente da dimostrare e ottenuto concatendo regole. Un **theorem prover** è un programma che, dato un sequente, prova a costruirne la dimosrazione. Nello specifico la versione bottom-up mantiene un insieme di sequenti da dimostrare, detti "goal", e ad ogni iterazione:

1. si sceglie un nuovo goal;
2. lo si scompone applicando una regola "al contrario";
3. si aggiugnono i nuovi goal al working-set.

Nella maggior parte dei casi la scelta della regola è non-deterministica.

Logica lineare

1 Introduzione

La logica lineare è una logica avente le seguenti caratteristiche:

- è generalmente proibita la duplicazione o l'eliminazione di formule, tutte le formule devono essere usate una e una sola volta;
- la negazione è simmetrica e un'involuzione;
- 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.

Durante il proof searching bottom-up in logica lineare un'ulteriore fonte di non-determinismo è l'operazione di **splitting**.

$$\otimes \frac{\vdash \Delta', \phi' \qquad \vdash \Delta'', \phi''}{\vdash \Delta', \Delta'', \phi' \otimes \phi''}$$

La scomposizione dei moltiplicativi comporta il partizionamento del contesto in Δ' e Δ'' per continuare la dimostrazione.

Indice

2 Il calcolo

Calcolo dei vincoli

2 Il calcolo

Nel 2001 D.Pym e J.Harland propongono un algoritmo alternativo per gestire le risorse (nello specifico lo splitting) basato su vincoli booleani. Questi possono essere solo di due tipi:

- una certa formula è stata usata per dimostrare questo goal, e non può essere usata altrove;
- una certa formula non è stata usata per dimostrare questo goal, e deve essere utilizzata per un altro.

Se i vincoli sono soddisfacibili, allora la dimostrazione utilizza le formule in modo corretto.

Focusing e normalizzazione

2 Il calcolo

Al calcolo abbiamo applicato due classiche modifiche nell'ambito del proof searching:

Focusing e normalizzazione

Al calcolo abbiamo applicato due classiche modifiche nell'ambito del proof searching:

- la normalizzazione permette di ridurre il numero delle regole e di lavorare solo con sequenti one-sided

$$\Gamma \vdash \Delta \Rightarrow \vdash \Delta$$

Focusing e normalizzazione

Al calcolo abbiamo applicato due classiche modifiche nell'ambito del proof searching:

- la normalizzazione permette di ridurre il numero delle regole e di lavorare solo con sequenti one-sided

$$\Gamma \vdash \Delta \Rightarrow \vdash \Delta$$

- il focusing suddivide la dimostrazione in due fasi che si alternano:
  - una fase (asincrona) in cui sono ammesse solo regole invertibili, cioè il cui ordine di applicazione non è importante;
  - una fase (sincrona) in cui ci si concentra su una formula e si continuano ad applicare regole non invertibili, con la possibilità di dover fare backtracking.

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

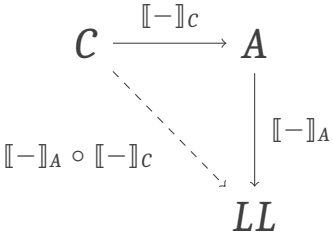
Correttezza

2 Il calcolo

Del 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, sfruttando le librerie per il CLP;

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

- un'implementazione in SWI-Prolog, sfruttando le librerie per il CLP;
- un generatore di test in OCaml;

Implementazione

2 Il calcolo

Per il calcolo di sopra sono state scritte:

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

Infine l'infrastruttura è stata gestita con Nix.

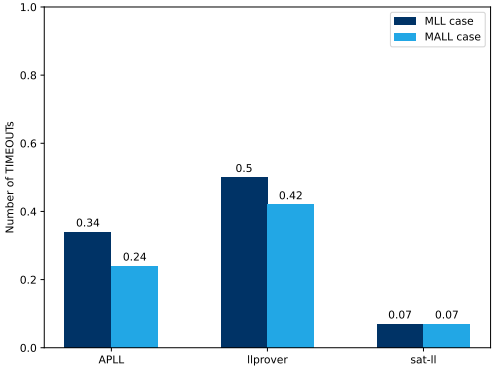
Indice

3 Conclusione

Risultati – caso moltiplicativo

Una volta implementato il prover lo si è confrontato con altri due prover:

- llprover (1997)
- APLL (2019)



Risultati – caso generale

Nel caso dei test esponenziali i risultati si livellano:

prover	timeouts	failures	successes	success rate	avg. (succ.)	avg. (tot.)
APLL	0	17	71	≈ 0.80	0.035 s	0.326 s
llprover	20	6	62	≈ 0.70	0.981 s	2.179 s
sat-ll	5	15	68	≈ 0.77	0.443 s	0.496 s

(a) Output per KLE-cbv

prover	timeouts	failures	successes	success rate	avg. (succ.)	avg. (tot.)
APLL	0	16	72	≈ 0.80	0.037 s	0.055 s
llprover	20	6	62	≈ 0.70	1.709 s	3.253 s
sat-ll	4	18	66	≈ 0.75	0.130 s	0.185 s

(b) Output per KLE-cbn

Fine.