



# 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. Ghentzen, 1934) è un formalismo per rappresentare un calcolo logico e strutturarne dimostrazioni. I concetti principali sono due:

- il sequente  $\Delta \vdash \Gamma$ , che esprime una implicazione tra la congiunzione di  $\Delta$  e la disgiunzione di  $\Gamma$ ;
- la regola, che descrive come si possono comporre i sequenti,

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

in questo caso se valgono  $\Pi'$  e  $\Pi''$ , allora vale  $\Pi$ .



# Calcolo dei sequenti cont'd

## 1 Introduzione

Una dimostrazione consiste in un albero avente come radice il sequente da dimostrare e ottenuto concatenando regole.

$$\begin{array}{c} \frac{A \frac{}{a, b \vdash b} \quad A \frac{}{a, b \vdash a}}{\wedge_R \frac{}{a, b \vdash b \wedge a}} \\ \wedge_L \frac{}{a \wedge b \vdash b \wedge a} \end{array}$$

Un theorem prover è un programma che, dato un sequente, prova a costruirne la dimostrazione applicando le regole di sopra. Nello specifico un theorem prover è detto bottom-up se parte dal sequente da dimostrare, e lo continua a scomporre applicando le regole “al contrario”.



# Logica lineare

## 1 Introduzione

La **logica lineare** (J.-Y. Girard, 1987) è una logica la cui caratteristica principale è che è generalmente proibita la duplicazione o l'eliminazione di formule durante il processo di dimostrazione, operazioni che altresì sono spesso lasciate implicito. Questa scelta porta ad alcune proprietà particolari, tra cui la duplicazione dei connettivi della logica classica secondo due interpretazioni diverse, una detta moltiplicativa ed l'altra additiva.

|          |               |              |              |
|----------|---------------|--------------|--------------|
| $\wedge$ |               | $\otimes$    | $\&$         |
| $\vee$   |               | $\wp$        | $\oplus$     |
| $\top$   | $\rightarrow$ | $\mathbf{1}$ | $\top$       |
| $\perp$  |               | $\perp$      | $\mathbf{0}$ |



# Splitting

## 1 Introduzione

Durante il proof searching bottom-up uno dei problemi principali è quello dello **splitting**.

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

La decomposizione del tensore (regola di qui sopra) comporta il partizionamento di  $\Delta$  in  $\Delta'$  e  $\Delta''$  per continuare la dimostrazione. Questa operazione può richiedere un numero esponenziale di tentativi.

Nel 2001 D.Pym e J.Harland propongono un metodo per gestire lo splitting utilizzando vincoli booleani generati a partire da espressioni associate a ogni singola formula.



# Indice

## 2 Il calcolo

► Introduzione

► **Il calcolo**

► Conclusione



# Focusing e normalizzazione

## 2 Il calcolo

Partendo dal calcolo citato sopra di D.Pym e J.Harland, abbiamo applicato due classiche modifiche nell'ambito del proof searching (i.e. normalizzazione e focusing), e abbiamo mostrato la correttezza del nuovo calcolo esibendo una traduzione ad un calcolo intermedio tale che il seguente diagramma commuta:

$$\begin{array}{ccc} C & \xrightarrow{\llbracket - \rrbracket_c} & A \\ & \searrow \llbracket - \rrbracket_A \circ \llbracket - \rrbracket_c & \downarrow \llbracket - \rrbracket_A \\ & & LL \end{array}$$





# Implementazione

## 2 Il calcolo

Per il calcolo di sopra:

- è stata fatta un'implementazione in SWI-Prolog;
- è stato scritto un generatore di test in OCaml;
- è stato architettato un sistema per il testing e il benchmarking in Python.

Infine l'infrastruttura per tutti i programmi è stata gestita con Nix.





# Indice

## 3 Conclusione

► Introduzione

► Il calcolo

► Conclusione



*Grazie per l'attenzione!*