# Proof Search in Propositional Linear Logic via Boolean Constraints Satisfaction

Martino D'Adda – 964827

21 giugno 2024

## 1 Ente presso cui è stato svolto il lavoro di stage

Il lavoro di tirocinio è stato di tipologia interna presso il dipartimento di Informatica dell'Università degli Studi di Milano, sotto la supervisione del docente Alberto Momigliano e il docente Camillo Fiorentini.

## 2 Contesto iniziale

Nell'ambito della dimostratori automatici bottom-up per la logica lineare, una delle principali fonti di complessità è un'operazione chiamata splitting. Questa richiede che ripetutamente durante la computazione della dimostrazione si debba partizionare il multiset di formule, portando ad una esplosione combinatoria che si aggiunge all'inerente complessità del problema del proof searching.

Il lavoro di tirocinio inizia da un articolo del 2001 di J. Harland e D. Pym [2] dove si propone un metodo alternativo per affrontare lo splitting affidandosi a vincoli booleani. I vincoli vengono generati in modo tale che se questi sono soddisfacibili, allora la dimostrazione è corretta, e i multiset di formule non vengono mai effettivamente partizionare. In questo la complessità viene spostata dalla generazione della partizione corretta, alla ricerca di un assegnamento per vincoli booleani.

## 3 Obiettivi del lavoro

L'obiettivo principale del lavoro è stato mostrare una implementazione di un prover basato sul calcolo di sopra e, una volta fatto questo, valutarne la sua efficacia anche rispetto ad altri prover già esistenti.

## 4 Descrizione del lavoro svolto

Inizialmente è stato implementato un prover basato direttamente sul calcolo di [2], ma questo si è rivelato inefficiente. Successivamente sono state applicate alcune modifiche ormai assodate nell'ambito del theorem proving quali la normalizzazione e il focusing [1]. Di questo nuovo calcolo è stata prima dimostrata la correttezza, esibendo una traduzione verso il calcolo triadico di [1]. Infine la nuova implementazione è stata confrontata con due altri prover: APLL [4] and llprover [3].

# 5 Tecnologie coinvolte

Sono state coinvolte le seguenti tecnologie:

- il linguaggio SWI-Prolog per la scrittura del prover, con particolare enfasi sulle sue librerie per il constraint logic programming (CLP);
- il linguaggio OCaML per la scrittura di un generatore di formule randomiche;
- il linguaggio Python e Jupyter Notebook per l'infrastruttura necessaria per comparare diversi prover;
- il linguaggio Nix (flake) e bash per l'automazione dei processi di compilazione e la generazione di ambienti di sviluppo riproducibili.

# 6 Competenze e risultati raggiunti

I benchmark hanno evidenziato come l'utilizzo dei vincoli booleani, assieme al focusing e alla normalizzazione, permettano di ottenere risultati competitivi nell'ambito della logica lineare moltiplicativa.

Il lavoro di tirocinio mi ha dato l'opportunità di avvicinarmi all'ambito della dimostrazione automatica, e a familiarizzarmi con la notazione del calcolo dei sequenti. Inoltre l'utilizzo di Prolog per l'implementazione mi ha permesso di esplorare diversi ambiti della programmazione logica, ad esempio le CFG e CLP.

Una delle problematiche principali si è rivelata essere la carenza di test senza esponenziali. Visto che l'obiettivo del prover era di mostrare un metodo per gestire efficientemente i moltiplicativi, ma la quasi totalità dei dataset di formule di logica lineare è costituito da traduzioni di formule intuizioniste o di problemi per reti di Petri; entrambi casi caratterizzati da un alto numero di esponenziali. La soluzione trovata è stato modificare un generatore randomico di formule affinché generasse casi senza esponenziali. Anche la carenza di prover mantenuti si è rivelato essere un problema, infatti un grande numero di quelli trovati citati si sono rivelati essere scomparsi o non funzionanti.

# Riferimenti bibliografici

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992.

[2] James Harland and David J. Pym. Resource-distribution via boolean constraints. *ACM Trans. Comput. Log.*, 4(1):56–90, 2003.

[3] Naoyuki Tamura. A linear logic prover (llprover). `https://cspsat.gitlab.io/llprover/`. Accessed: 2024-03-20.

[4] Jui-Hsuan Wu. A linear logic prover implemented in ocaml. `https://github.com/wujuihsuan2016/LL_prover`. Accessed: 2024-03-20.