Linear Logical Algorithms

Frank Pfenning

Carnegie Mellon University

Workshop on Programming Logics

in memory of Harald Ganzinger

Saarbrücken, Germany, June 3–4, 2005

Acknowledgments

- Harald Ganzinger
- Joint work with Iliano Cervesato, Dave Walker, Kevin Watkins, Pablo López, Jeff Polakow
- Supported by ONR N00014-04-1-0724, *Distributed System Security via Logical Frameworks*
- `http://www.cs.cmu.edu/~self`
- *Work in progress!*

Overview

- Logical Algorithms
- Beyond Pure Saturation
- Linear Logical Algorithms
- Logical Foundation
- Operational Interpretation
- Conclusion

Bottom-Up Logic Programming

- Describe algorithm with a set of rules
- Example: transitive closure R of relation E

$$\frac{E(x,y)}{R(x,y)}\ \mathsf{R_1} \qquad\qquad \frac{R(x,y) \quad R(y,z)}{R(x,z)}\ \mathsf{R_2}$$

- Start with database of *ground facts*
- Apply rules until *saturated*: no inference adds any new facts

Complexity Analysis

- Based on *prefix firings* [McAllester'99,'02]
- Example: transitive closure R of relation E

$$\frac{E(x,y)}{R(x,y)}\ \mathsf{R}_1 \qquad\qquad \frac{R(x,y) \quad R(y,z)}{R(x,z)}\ \mathsf{R}_2$$

- $O(n^2)$ firings of $R(x,y)$ in R_2
- $O(n)$ firings of $R(y,z)$ if y fixed
- Overall complexity $O(n^3)$

Alternative Transitive Closure

- Single step rules

$$\frac{E(x,y)}{R(x,y)}\ \mathsf{S}_1 \qquad\qquad \frac{E(x,y) \quad R(y,z)}{R(x,z)}\ \mathsf{S}_2$$

- $O(e)$ firings for $E(x,y)$ in S_2
- $O(n)$ firings for $R(y,z)$ if y fixed
- $O(e\,n)$ overall complexity
- More efficient for sparse graphs
- Order of premisses matters

Bottom-Up vs. Top-Down

- Even test, bottom-up, forward chaining

$$\frac{\mathsf{ev}(\mathsf{s}(\mathsf{s}(x)))}{\mathsf{ev}(x)}\ \mathsf{F}_1$$

  - Initialize with $\mathsf{ev}(n)$ for given n
  - Saturate database under F_1
  - Check if $\mathsf{ev}(0)$ in database

- Even test, top-down, backward chaining

$$\frac{}{\mathsf{even}(0)}\ \mathsf{E}_0 \qquad \frac{\mathsf{even}(x)}{\mathsf{even}(\mathsf{s}(\mathsf{s}(x)))}\ \mathsf{E}_1$$

Limitations of Pure Saturation

- Other examples
  - CKY parsing
  - Program analysis (liveness, dataflow)
  - Unification
  - Congruence closure
  - Type inference
- Many algorithms cannot be described naturally
- Both efficiency and functionality at issue

Beyond Pure Saturation

- Introduce *deletion* and *priorities* [Ganzinger & McAllester'01,'02]
- Motivated from saturation-based theorem proving
  - Deletion from redundancy elimination
  - Priorities from ordering constraints
- Deletion introduces *don't-care non-determinism*
- Priorities compensate
  - Sometimes, for complexity (efficiency) only
  - Sometimes, for functional correctness
- No clear logical origin

Checking Bipartiteness

- Color adjacent nodes with different colors A, B
- Label rules (R, n) with name R and priority n

$$\frac{E(x,y)}{E(y,x)}\ (\mathsf{R}_1, 1) \qquad \frac{\mathsf{lab}(x,A) \quad E(x,y)}{\mathsf{lab}(y,B)}\ (\mathsf{R}_2, 1) \qquad \frac{\mathsf{lab}(x,B) \quad E(x,y)}{\mathsf{lab}(y,A)}\ (\mathsf{R}_3, 1)$$

$$\frac{\mathsf{lab}(x,c)}{\mathbf{del}(\mathsf{unlab}(x))}\ (\mathsf{R}_4, 1) \qquad \frac{\mathsf{unlab}(x)}{\mathsf{lab}(x,A)}\ (\mathsf{R}_5, 2)$$

- Use R_5 only none of R_1 – R_4 apply
- Rule R_4 permanently deletes $\mathsf{unlab}(x)$

Operational Interpretation

- Initially, $\mathsf{unlab}(x)$ for every node x
- Saturate
- Test, if there is x such that $\mathsf{lab}(x, A)$ and $\mathsf{lab}(x, B)$
  - If yes, graph is not bipartite
  - If no, graph is bipartite
- Complexity $O(e + n)$

Linear Logical Algorithms

- Eliminate explicit deletion and priorities
- Generalize underlying logic to be *linear* [Girard'87]
  - *Ephemeral* facts consumed by rule application
  - *Persistent* facts remain
- Implemented in logic programming language LOLLIMON [López,Pf,Polakow,Watkins'05]
- Combines bottom-up and top-down inference
- Termination requires *saturation* and *quiescence*
- Complexity analysis?

Checking Bipartiteness, Revisited

- Color *ephemeral* facts red, persistent facts black
- Rules R_1 – R_3 unchanged (without priorities)
- Rule R_4 consumes ephemeral $\mathsf{unlab}(x)$

$$\frac{E(x,y)}{E(y,x)}\ \mathsf{R}_1 \qquad \frac{\mathsf{lab}(x,A) \quad E(x,y)}{\mathsf{lab}(y,B)}\ \mathsf{R}_2 \qquad \frac{\mathsf{lab}(x,B) \quad E(x,y)}{\mathsf{lab}(y,A)}\ \mathsf{R}_3 \qquad \frac{\mathsf{lab}(x,c) \quad \mathsf{unlab}(x)}{\cdot}\ \mathsf{R}_4$$

- Rule R_5 would be incorrect

$$\frac{\mathsf{unlab}(x)}{\mathsf{lab}(x,A)}\ \mathsf{R}_5 \mathsf{X}$$

Operational Interpretation

- Initially, ephemeral $\mathsf{unlab}(x)$ for every node x
- Would like to implement

1. Pick (and consume) $\mathsf{unlab}(x)$
   - If there is no such x, graph is bipartite; stop
2. Color x with A
3. Run rules R_1 – R_4 to saturation
4. Test, if there is x such that $\mathsf{lab}(x, A)$ and $\mathsf{lab}(x, B)$
   - If yes, graph is not bipartite; stop
   - If not, goto (1)

Top-Down Logic Programming

- Use the following rules *backwards*

$$\dfrac{\text{unlab}(x) \quad \dfrac{\dfrac{}{\text{lab}(x,A)}}{\vdots}\{\text{notbipartite}\}}{\text{notbipartite}}\,\text{R}_5 \qquad \dfrac{\text{lab}(x,A) \quad \text{lab}(x,B)}{\text{notbipartite}}\,\text{R}_6$$

- $\text{lab}(x, A)$ is new fact in right subderivation of R_5
- $\{\text{notbipartite}\}$ indicates that we saturate facts before attempting to solve subgoal
- R_6 makes success criterion explicit

Checking Even, Revisited

- Use following E_0 backwards
- Use E_1 only forwards (conclusion $\{\ldots\}$)

$$\dfrac{\dfrac{}{\mathsf{ev}(n)}\quad\vdots\quad\{\mathsf{ev}(0)\}}{\mathsf{even}(n)}\,\mathsf{E}_0 \qquad \dfrac{\mathsf{ev}(\mathsf{s}(\mathsf{s}(n)))}{\{\mathsf{ev}(n)\}}\,\mathsf{E}_1$$

- Integrates initialization, success criterion
- Could make all $\mathsf{ev}(\_)$ ephemeral

Bellman-Ford Algorithm

- Concurrent algorithm for shortest path
- Assume here no negative weight cycles
- Initialize with ephemeral $\mathsf{vertex}(x)$ for every node x
- Starting rule (call with source node x_0)

$$\dfrac{\mathsf{vertex}(x_0) \qquad \begin{array}{c}\overline{\mathsf{dist}(x_0, 0)}\\ \vdots \\ \{\top\}\end{array}}{\mathsf{source}(x_0)}$$

Bellman-Ford Algorithm, Continued

- Persistent facts $\mathsf{edge}(x, w, y)$ for edge from x to y with weight w
- Propagation rules

$$\frac{\mathsf{edge}(x,w,y) \quad \mathsf{dist}(x,d) \quad \mathsf{vertex}(y)}{\{\mathsf{dist}(x,d), \mathsf{dist}(y,d+w)\}}$$

$$\frac{\mathsf{edge}(x,w,y) \quad \mathsf{dist}(x,d) \quad \mathsf{dist}(y,e) \quad d+w<e}{\{\mathsf{dist}(x,d), \mathsf{dist}(y,d+w)\}}$$

Logical Foundation

- Logical foundation
  - Intuitionistic linear logic (standard)
  - Lax modality $\{\_\}$ (standard)
  - Novel combination [Cervesato,Pf,Walker,Watkins'02]
- Fragment with tractable operational semantics

$$
\begin{array}{lrcl}
\text{Asynch Types} & A & ::= & P \mid A_1 \rightarrow A_2 \mid \forall x.\, A(x) \mid A_1 \multimap A_2 \\
 & & & \mid A_1 \,\&\, A_2 \mid \top \mid \{S\} \\
\text{Synch Types} & S & ::= & A \mid !A \mid \exists x.\, S(x) \mid S_1 \otimes S_2 \mid 1
\end{array}
$$

Checking Even, Revisited

- Even, linear forward chaining

$$\mathsf{even}(x) \circ\!\!- (\mathsf{ev}(x) \multimap \{\mathsf{ev}(0)\}).$$
$$\mathsf{ev}(\mathsf{s}(\mathsf{s}(x))) \multimap \{\mathsf{ev}(x)\}.$$

- Even, backward chaining (top-down)

$$\mathsf{even}(0).$$
$$\mathsf{even}(\mathsf{s}(\mathsf{s}(x))) \leftarrow \mathsf{even}(x).$$

Bellman-Ford, Revisited

- $A \rightarrow$ _ means A persistent (unrestricted impl)
- $A \multimap$ _ means A ephemeral (linear implication)

$$\mathsf{edge}(x, w, y) \rightarrow \mathsf{dist}(x, d) \multimap \mathsf{dist}(y, e) \multimap d + w < e \rightarrow \{\mathsf{dist}(x, d) \otimes \mathsf{dist}(y, e)\}.$$

$$\mathsf{edge}(x, w, y) \rightarrow \mathsf{dist}(x, d) \multimap \mathsf{vertex}(y) \multimap \{\mathsf{dist}(x, d) \otimes \mathsf{dist}(y, d + w)\}.$$

$$\mathsf{source}(x_0) \circ\!\!- \mathsf{vertex}(x_0) \circ\!\!- (\mathsf{dist}(x_0, 0) \multimap \{\top\}).$$

Lax Modality

- Also known as *strong monad*
- Logical laws

$$\text{unit:} \quad \vdash A \multimap \{A\}$$
$$\text{bind:} \quad \vdash \{A\} \multimap (A \multimap \{B\}) \multimap \{B\}$$

- Formulated more appropriately via inference rules
- Widely used in functional programming
- New in logic programming

Operational Interpretation

- Goal $\{A\}$ (law $A \multimap \{A\}$)
  - Suspend backward chaining of $\{A\}$
  - Forward chain to saturation
  - Resume backward chaining to solve goal A
- Clause with head $\{A\}$ (law $\{A\} \multimap (A \multimap \{B\}) \multimap \{B\}$)
  - Use clause only during forward chaining
  - Solve preconditions (usually: matching against facts)
  - Assert A into database
  - Continue forward chaining

Operational Interpretation, Ctd.

- Outside monad
  - Backchaining, with backtracking
  - Terminated by success or failure
  - Conservative over Horn LP (Prolog)
  - Conservative over linear LP (Lolli) [Hodas & Miller'94]
- Inside monad
  - Forward chaining, committed choice
  - Terminated by saturation and quiescence
  - Concurrent semantics

Bipartiteness, Revisited

- Transcribe rules ($!A$ for persistent fact A)

$$E(x, y) \rightarrow \{!E(y, x)\}.$$

$$\mathsf{lab}(x, A) \rightarrow E(x, y) \rightarrow \{!\mathsf{lab}(y, B)\}.$$

$$\mathsf{lab}(x, B) \rightarrow E(x, y) \rightarrow \{!\mathsf{lab}(y, A)\}.$$

$$\mathsf{lab}(x, c) \rightarrow \mathsf{unlab}(x) \multimap \{1\}.$$

$$\mathsf{notbipartite} \leftarrow \mathsf{lab}(x, A) \leftarrow \mathsf{lab}(x, B) \circ\!\!- \top.$$

$$\mathsf{notbipartite} \circ\!\!- \mathsf{unlab}(x) \circ\!\!- (\mathsf{lab}(x, A) \rightarrow \{\mathsf{notbipartite}\}).$$

- Initialize with ephemeral $\mathsf{unlab}(x)$ for every x

Summary

- Linear Logical Algorithms
  - Combination of top-down and bottom-up logic programming
  - Separated by lax modality (monad)
  - Linearity to model deletion
  - Backward chaining to model priorities/phases
  - Rich computational model
  - Derived from Concurrent Logical Framework (CLF) [Cervesato,Pf,Walker,Watkins'02,'04]
- *Elegant logical foundation!*

Future Work

- More efficient implementation
  - Unification vs. hash-consing?
  - Saturation vs. quiescence?
- Complexity analysis
  - Prefix firings
  - Linearity?
  - Stated invariants?
- Advanced termination criteria
  - Sets of states/model-checking