Hint Selection and Prioritization

Robert Veroff Josef Urban Michael Kinyon

U. New Mexico Czech Tech U. U. Denver

AITP 2019, Obergurgl, Austria, 9 April 2019

ATP and the Working Mathematician

Interactive Theorem Provers (ITPs) are great for assisting mathematicians in formalization, proof checking, and so on.

But for the discovery of *new* mathematics, it is often helpful to work directly with *automated* theorem provers (ATPs). This is especially true in research areas in which there are open problems which have first order formulations, that is, the goal reduces to proving some set of first-order clauses is unsatisfiable.

Since about the turn of the millenium, most of my work has used ATPs, especially PROVER9, in *quasigroup theory* and *semigroup theory* with some success.

Given Clauses

A very oversimplified form of the Given Clause Algorithm

```
while (no proof found)
{
    select a clause as given
    move it to the usable list
    apply inference rules to the
       given clause & other usable clauses
    process newly inferred clauses
}
```

Variants of this (“Otter loop” vs “Discount loop”) depend on whether or not clauses can be used for rewriting before they are selected as given. For our purposes here, this distinction is not important.

Given Clause Selection

It's all about the given clause. – Bill McCune

The success of a search depends *heavily* on how given clauses are selected. Strategies include:

- lightest first: weighting based on symbol count
- user-defined weighting patterns
- oldest first: pick clause that has waited the longest
- attribute-based restrictions (e.g., set of support)
- heuristic combinations of the above
- model-based selection (semantic guidance)
- subsumption-based selection (e.g., hints)
- statistical methods (e.g., machine learning)

Hints

Our focus is on *hints*.

- A hint is a user-supplied clause. An inferred clause *matches* a hint if subsumes it.
- Bias the selection of given clauses toward hint matchers.
- Guides the search without overly constraining it.

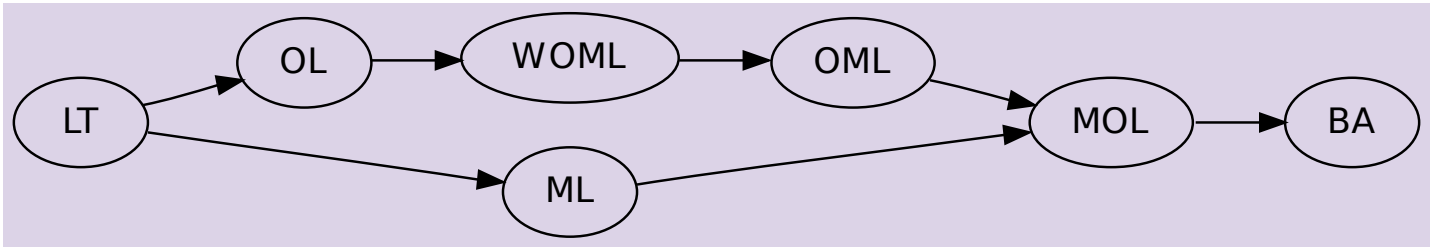
But where do the hints themselves come from?

Proof sketches

- In principle, the expert mathematician user can supply hints based on their prior knowledge of the area.
- In practice, hints come from proofs of related theorems.
  - related results in the same theory
  - the same result in slightly different theories
- Idea: the desired proof and the already found proofs will often share many of the same lemmas.
- Typically these are theorems we have proved ourselves as part of the process of proving the target theorem.
- Proving target theorems with extra assumptions and then iteratively eliminating them is an especially effective source of hints.

Where Do We Get Extra Assumptions?

Example: Lattice Theory Hierarchy



$$\mathcal{LT} = \text{lattice theory}$$
$$\mathcal{OL} = \text{ortholattices}$$
$$\mathcal{WOML} = \text{weakly orthomodular lattices}$$
$$\mathcal{OML} = \text{orthomodular lattices}$$
$$\mathcal{MOL} = \text{modular ortholattices}$$
$$\mathcal{BA} = \text{boolean algebras}$$

The AIM Project

The AIM Project involves a (huge!) hierarchy of varieties and extra assumptions, and since it is the part of all this I am involved in, let me pause the general discussion of hints to talk about it briefly.

This is not exactly a mathematics talk, so I will sidestep the full mathematical background of AIM and describe its essential features by looking at the parts of a generic Prover9 AIM input file. (The advantage over other systems' input conventions is that Prover9 input is much easier for mathematicians to read!)

Loops

```
% loop axioms
1 * x = x. x * 1 = x.
x \ (x * y) = y. x * (x \ y) = y.
(x * y) / y = x. (x / y) * y = x.
```

- Q is a *loop* if
  - $\forall a, b \in Q$, the equations $ax = b$, $ya = b$ have unique solutions $x = a\backslash b$, $y = b/a$ in Q
  - $\exists$ identity element $1 \in Q$: $1x = x = x1$
- Multiplication tables of loops $=$ reduced latin squares
- Can be viewed as having *three* binary operations, the multiplication and two divisions
- We sometimes include cancellation laws in the input to move things along although it is not necessary

Inner Mappings

```
% inner mappings
T(x,y) = y \ (x * y).
L(x,y,z) = (z * y) \ (z * (y * x)).
R(x,y,z) = ((x * y) * z) / (y * z).
```

- For each y, z, each of these is a permutation (in x) that fixes $x = 1$.
- The T_y's measure noncommutativity. (They are the usual conjugations in the group case.)
- The $L_{y,z}$'s and $R_{y,z}$'s measure nonassociativity
- The *inner mapping group* $Inn(Q)$ is the group generated by all T_y, $L_{y,z}$, $R_{y,z}$

AIM

```
% AIM = Abelian Inner Mappings
T(T(x,y),z) = T(T(x,z),y).
L(T(x,y),z,u) = T(L(x,z,u),y).
R(T(x,y),z,u) = T(R(x,z,u),y).
L(L(x,y,z),u,w) = L(L(x,u,w),y,z).
L(R(x,y,z),u,w) = R(L(x,u,w),y,z).
R(R(x,y,z),u,w) = R(R(x,u,w),y,z).
```

- These express equationally the postulate that $Inn(Q)$ is an abelian group.
- In groups, this is equivalent to being nilpotent of class 2.
- These equations introduce a lot of symmetry in the problem. Rule of thumb: highly symmetric problems (e.g., commutativity) $=$ big search spaces.

Commutators and Associators

```
% commutators and associators
K(x,y) = (y * x) \ (x * y).
a(x,y,z) = (x * (y * z)) \ ((x * y) * z).
```

- These are just conventions in the literature
- Commutators are not as closely tied to conjugation as in group case
- Similarly, these associators are not as closely tied to the $L_{y,z}$'s and $R_{y,z}$'s
- In retrospect, other definitions may have been more suitable
- But these just add to the challenge :-)

Goals

```
% Goals
K(a(x,y,z),u) = 1                    # label("Ka").
a(K(x,y),z,u) = 1                    # label("aK1").
a(x,K(y,z),u) = 1                    # label("aK2").
a(x,y,K(z,u)) = 1                    # label("aK3").
a(a(x,y,z),u,w) = 1                  # label("aa1").
a(x,a(y,z,u),w) = 1                  # label("aa2").
a(x,y,a(z,u,w)) = 1                  # label("aa3").
```

- These are 7 of the 8 identities which express the assertion that Q is nilpotent of class 2
- The 8th equation $K(x, K(y, z)) = 1$ is false in general
- The AIM Conjecture, the above 7 goals for general AIM loops, is still open!

Dependencies

It turns out:

$$(\text{Ka}) \iff (\text{aK1}) \iff (\text{aK2}) \iff (\text{aK3})$$

and

$$(\text{aa1}) \iff (\text{aa2}) \iff (\text{aa3})$$

Bob got proofs of most of these implications in 2012. The proof that (aK2) implies something else was not found until 2016, about six months after the first AITP conference.

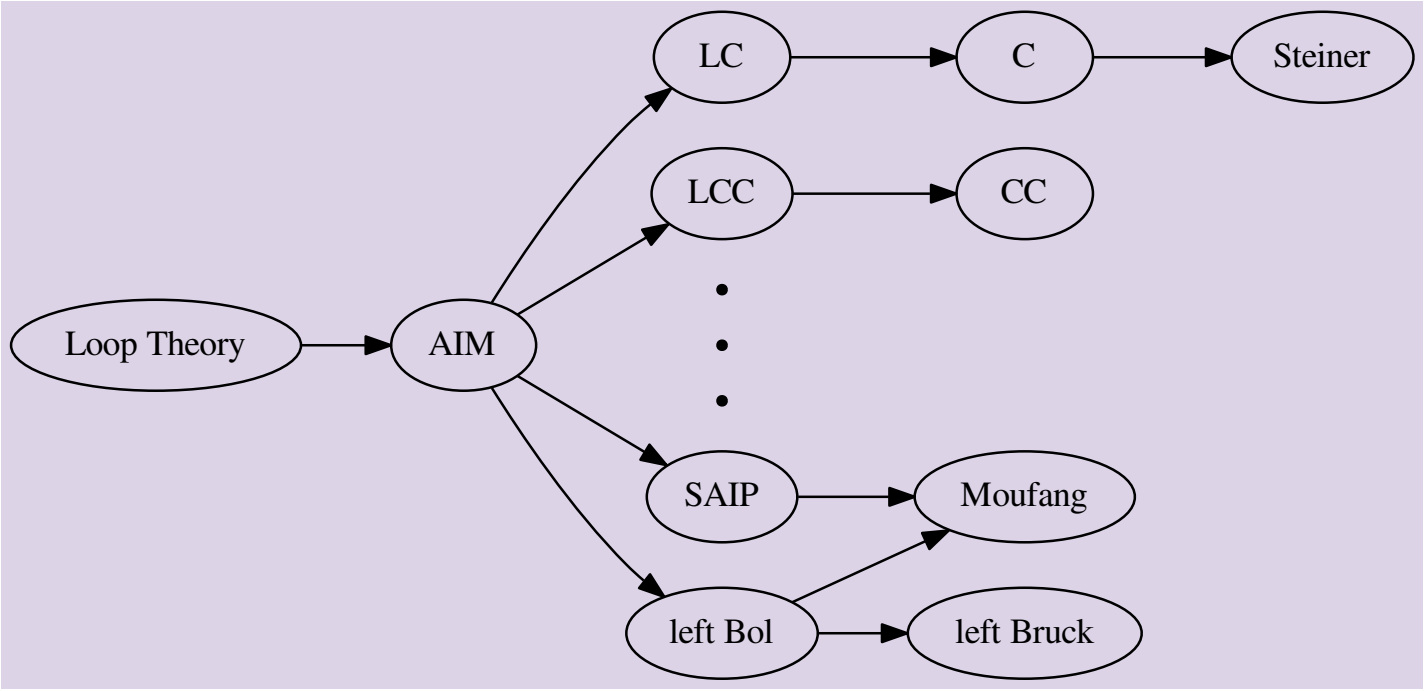
Back to Extra Assumptions

Getting back to our main theme, we can try to find proofs [of the AIM Conjecture] in the presence of extra assumptions. One source of such assumptions is to consider classes [of loops] which actually interest expert users [loop theorists].

- Moufang loops (like the nonzero octonions): $(xy)(zx) = x((yz)x)$
- C-loops: $((xy)y)z = x(y(yz))$
- Automorphic loops: every inner mapping is an automorphism
- and there are *many* others

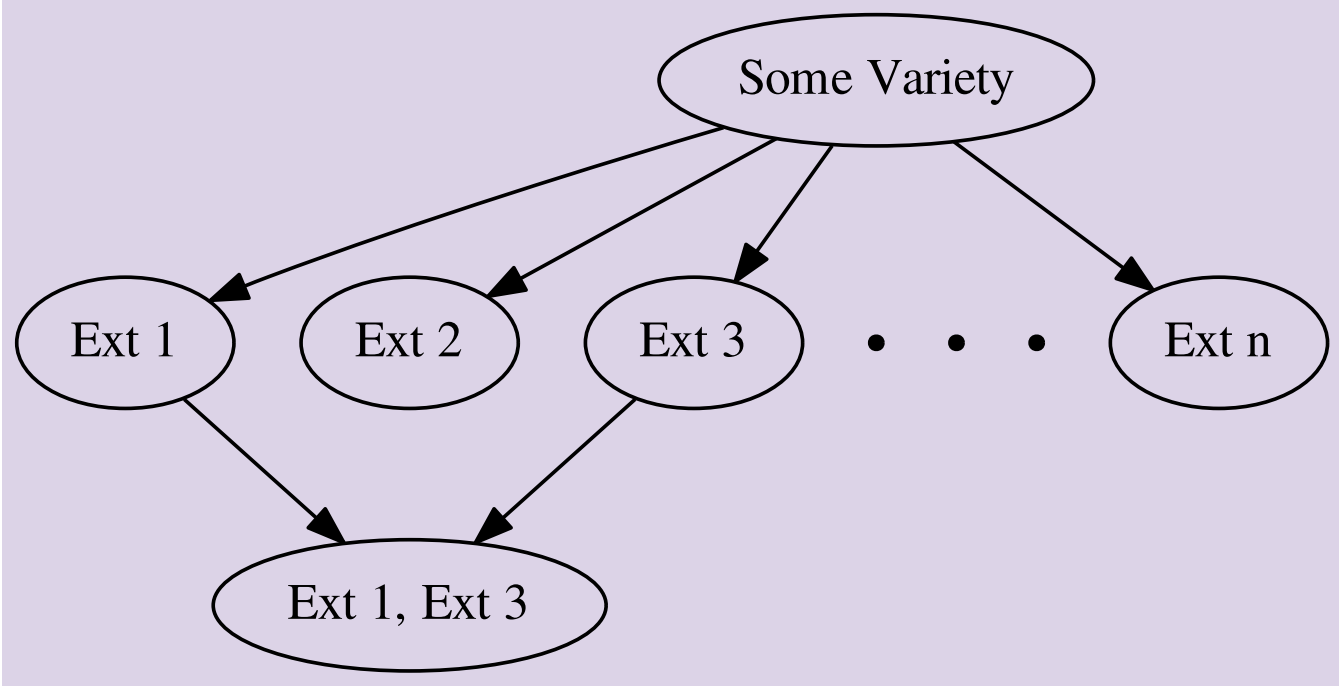
Loop hierarchy

A *tiny* fragment, presented in the AIM context.



Extensions

Work in extensions (varieties with extra assumptions), find proofs, use those proofs as hints, work our way up by eliminating the extra assumptions.



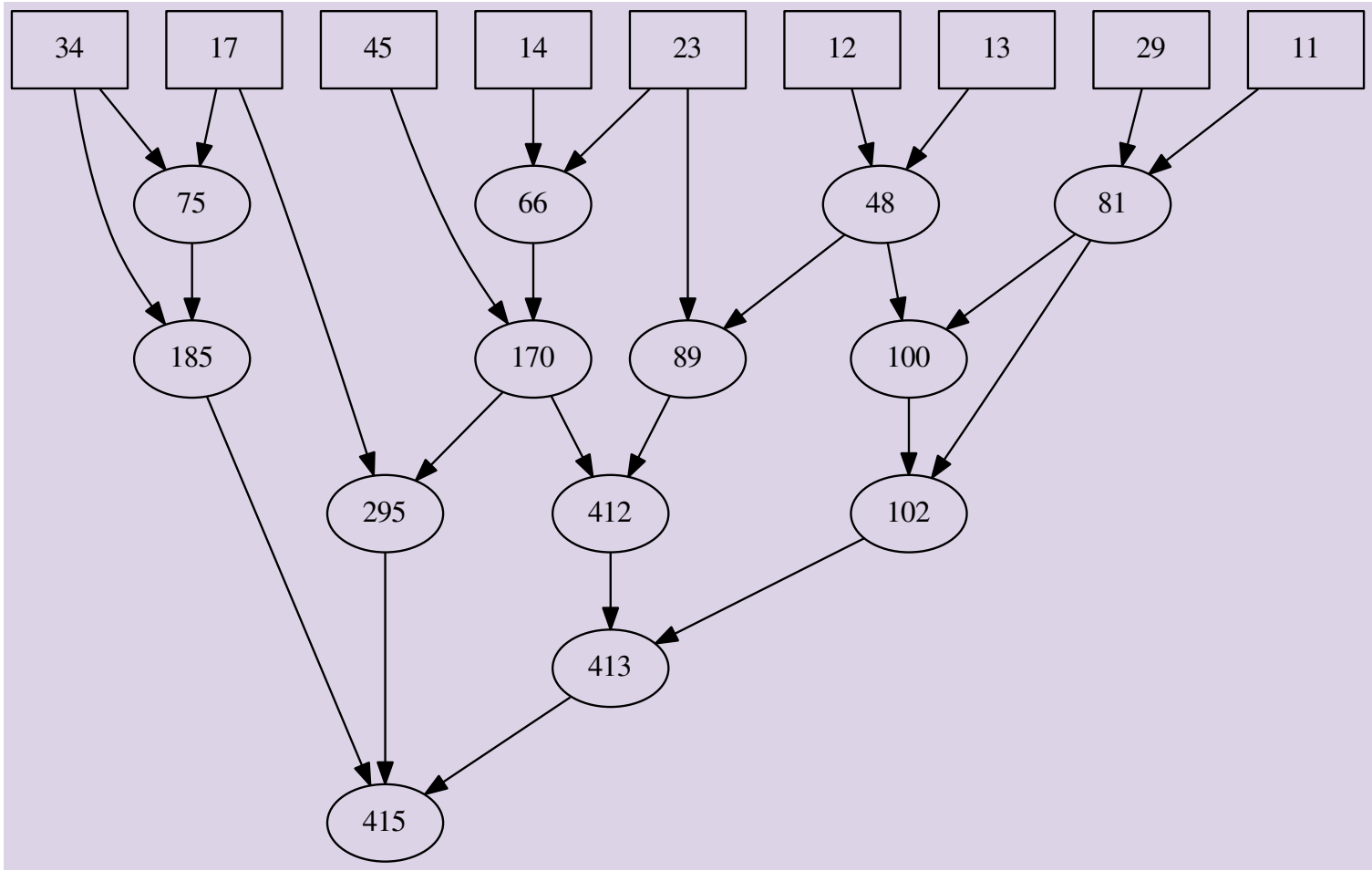
Example

Here is a proof (where what the clauses actually say is not important). Inferred clauses are followed by justifications.

```
11 P(11).   [assumption]
12 P(12).   [assumption]
13 P(13).   [assumption]
14 P(14).   [assumption]
17 P(17).   [assumption]
23 P(23).   [assumption]
29 P(29).   [assumption]
34 P(34).   [assumption]
45 P(45).   [assumption]
48 P(48).   [12,13]
66 P(66).   [14,23]
75 P(75).   [17,34]
81 P(81).   [11,29]
89 P(89).   [23,48]
100 P(100).   [48,81]
102 P(102).   [81,100]
170 P(170).   [45,66]
185 P(185).   [34,75]
295 P(295).   [17,170]
412 P(412).   [89,170]
413 P(413).   [102,412]
415 P(415).   [185,295,413]
```

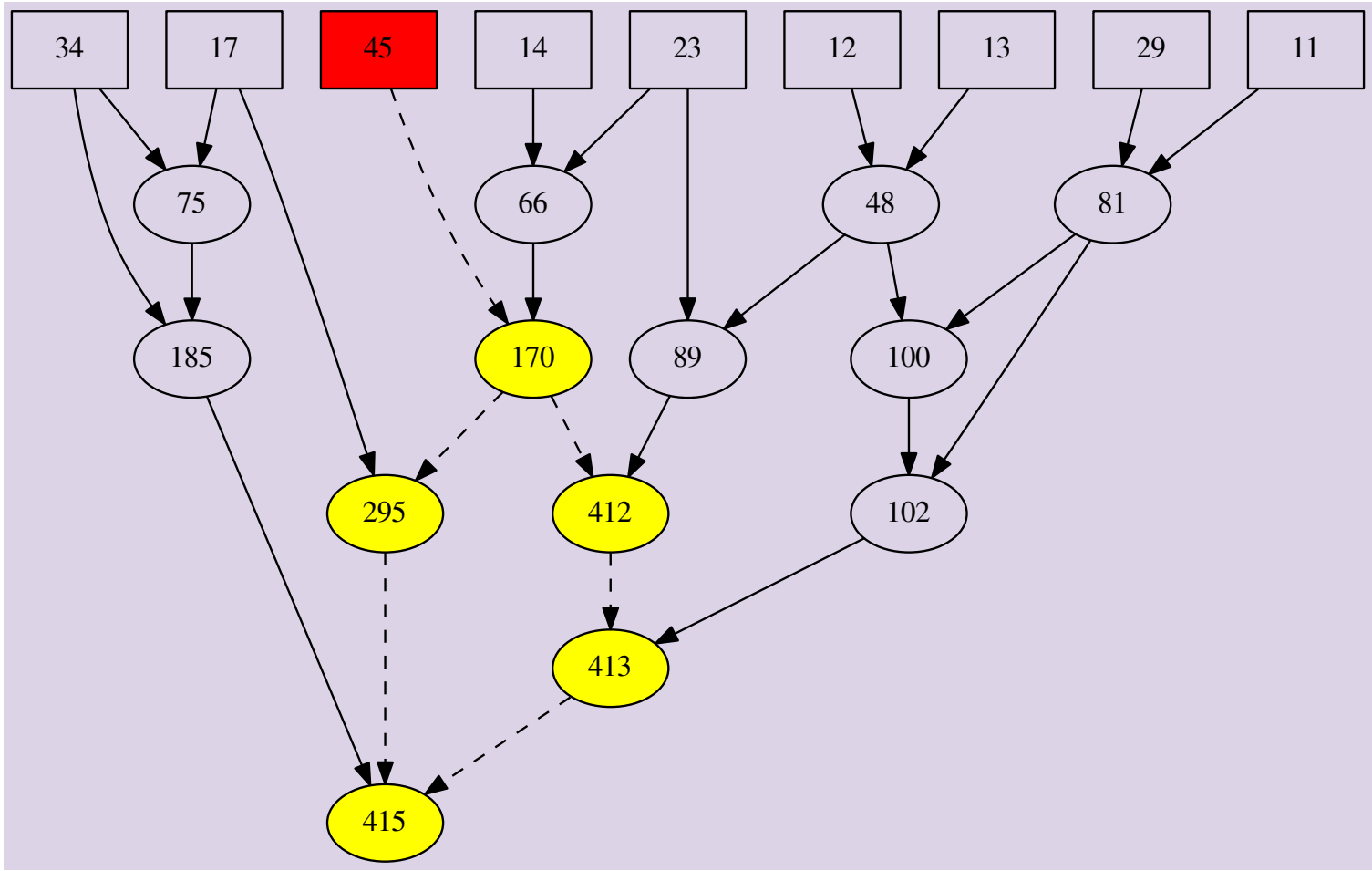
Inference Graph

Here is a graph of the clause dependencies for the same proof.

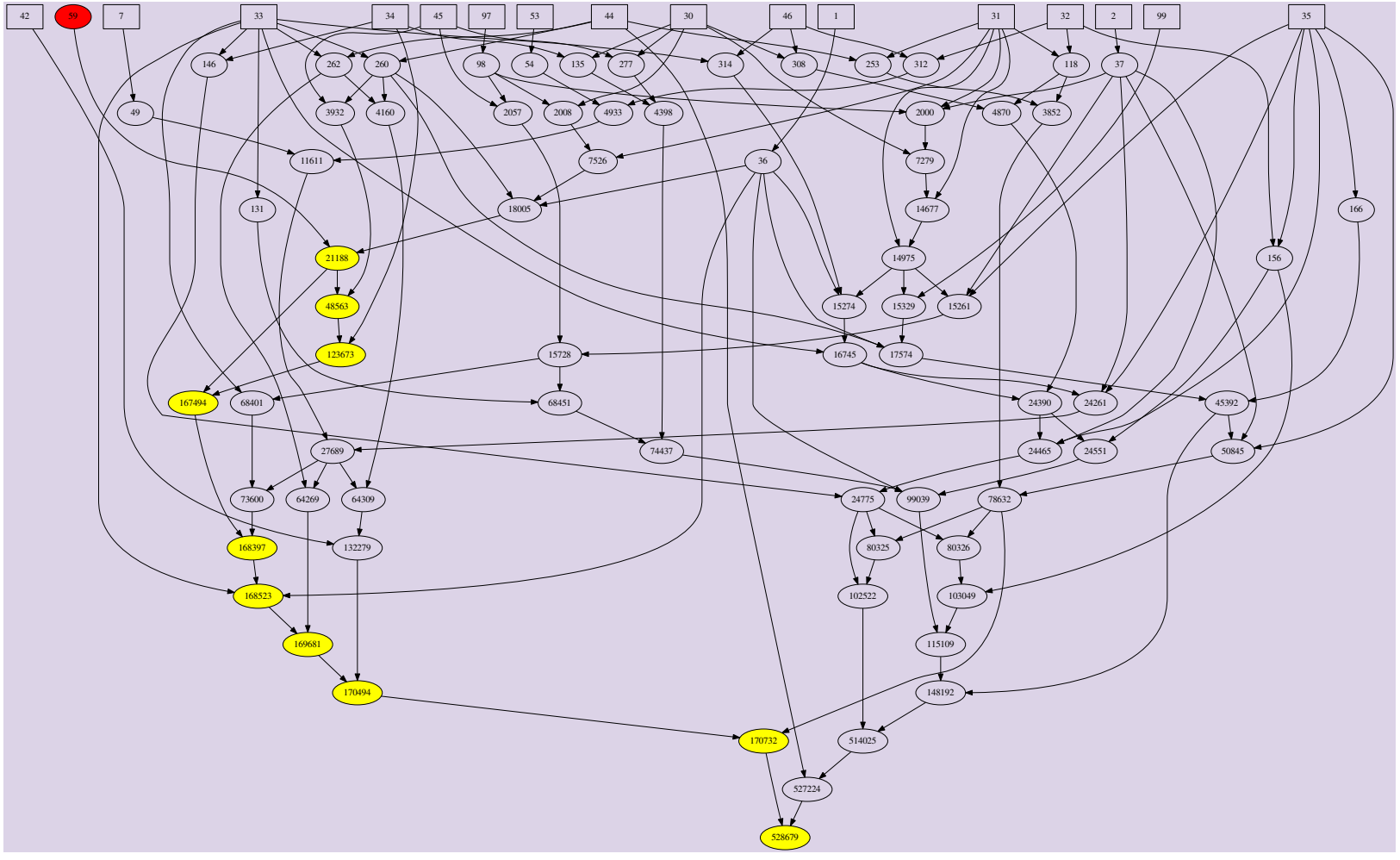


Eliminating an Assumption

Clauses depending on an extra assumption either are or are not in the target theory. If they are, they make useful hints (find another derivation of them). If not, including them as hints does no harm (they are never matched).



Larger Example



Hints Management

Problem:

- Large problems, like AIM, can lead to a *large* number of hints and hence a large number of hint matchers
- Too many hints can be both a distraction and inefficient (huge space of “high priority” clauses to select as given)

Ways to Cope:

- Take care in selecting sources for hints
  - tighten the definition of “related” problem
- Hint prioritization: prefer matchers of higher priority hints
  - prioritize sources (most relevant, most recent)
  - prioritize hints within a source
  - prioritize across sources

Running With Hints

Typically run with both a prioritized set of hints and an unprioritized set.

Selecting the next given clause:

- Select the clause that matches the highest priority hint.
- If none, select a clause that matches an unprioritized hint (typically by weight).
- If none, select any clause (by any classical strategy: weight, age, etc).

This approach allows us to have a larger set of potentially useful hints with less risk of the hints causing a distraction.

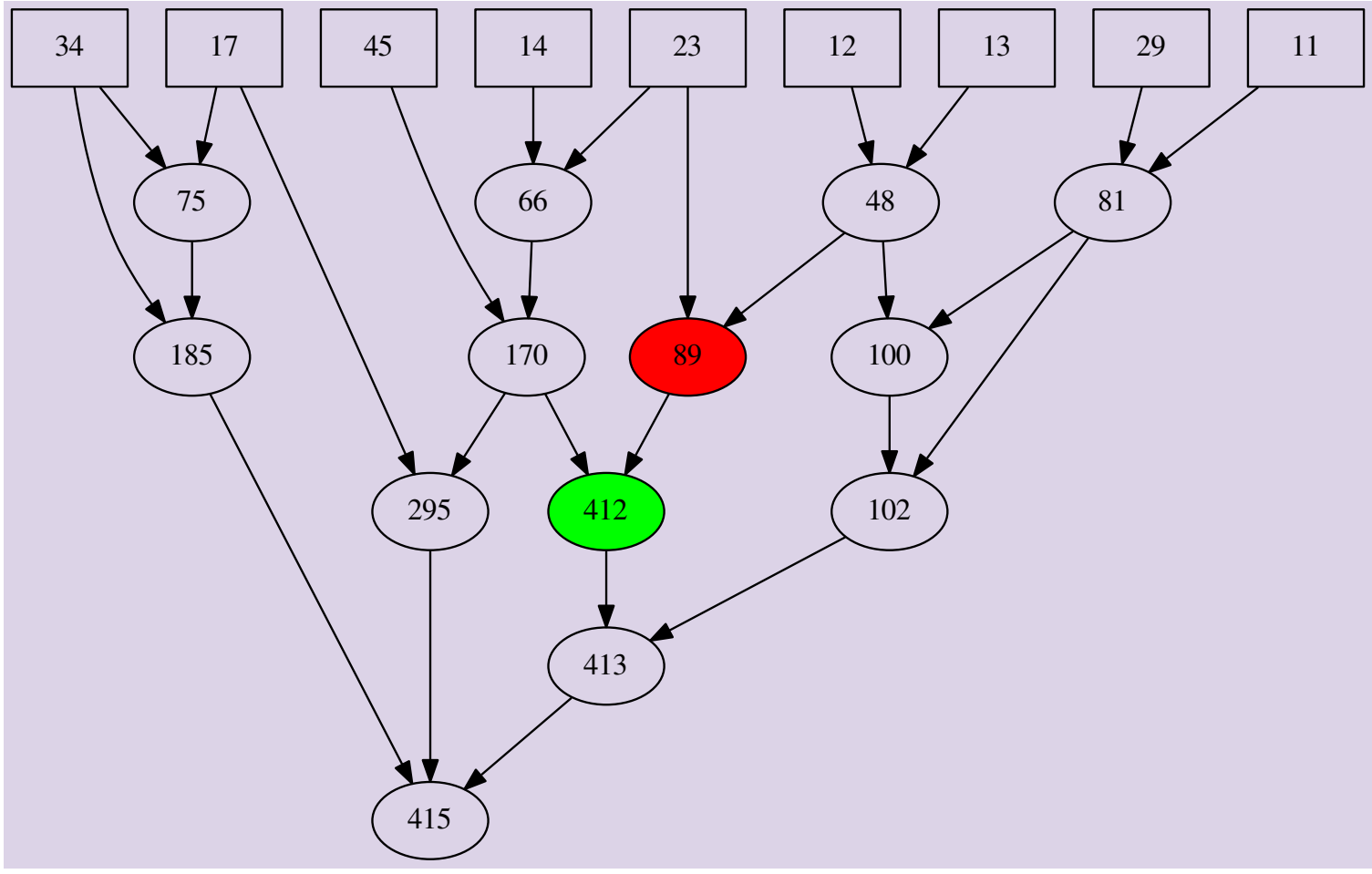
Recipe for a Difficult Problem

- Prove with a few extra assumptions $e_1 \ldots e_n$.
- For some e_i used in the proof, partition the proof clauses: those dependent on e_i in the proof (D) and those that are not dependent (ND).
- Prove without e_i, assuming the clauses in ND as lemmas and the previous proof as the highest priority hints.
- Prove again, without assuming the clauses in ND.
- Repeat, eliminating an extra assumption from the most recent proof in the same way.

One of our most recent results, and one of the best we have for the general AIM problem, was found this way:

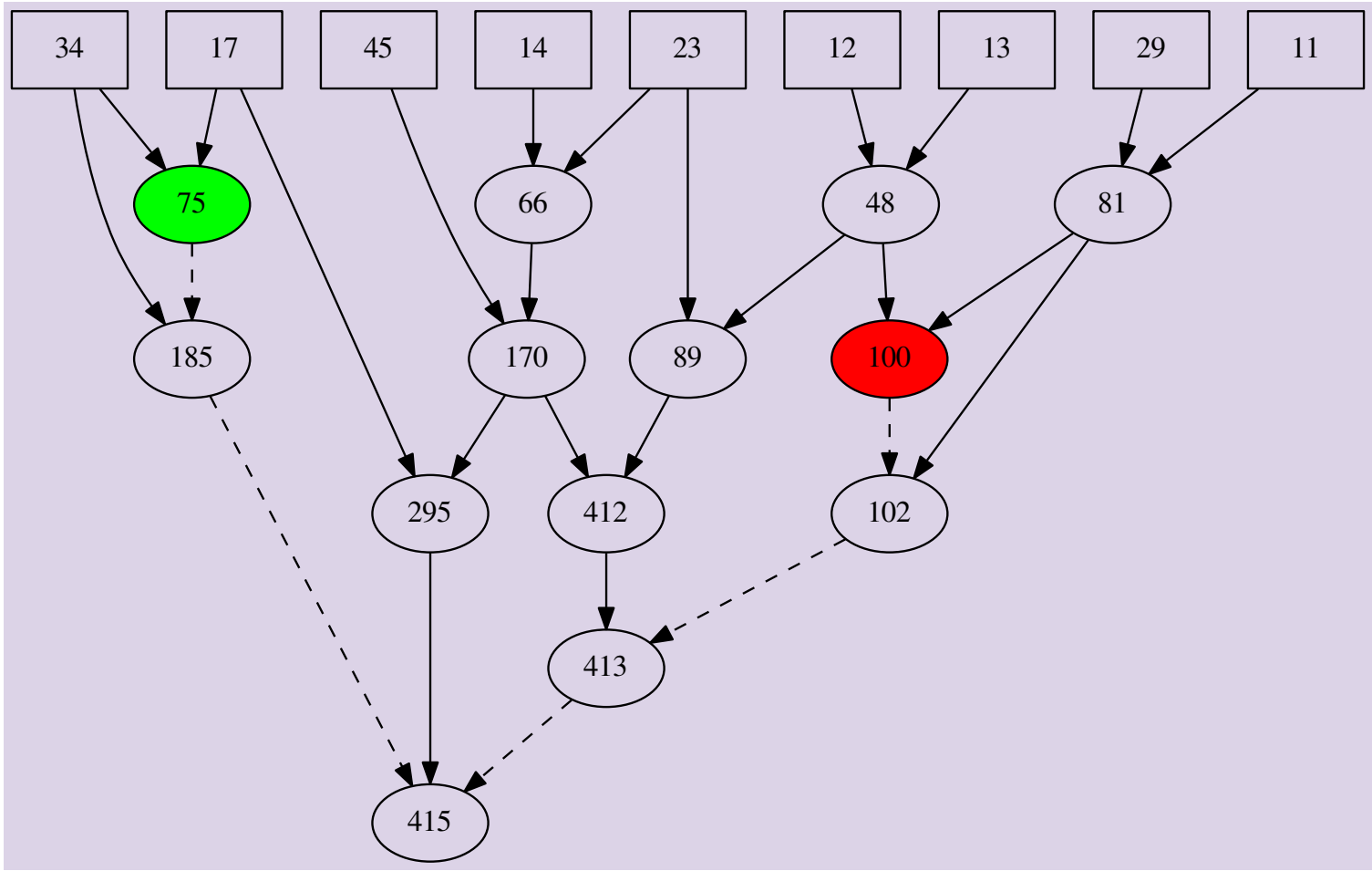
```
K(a(x,y,z),u) = a(x,y,K(z,u)).
```

Prioritizing by Clause Number



Prioritizing by Inference Distance

Meaning: distance to the empty clause or to an interesting derived clause



Inference Difference (BFS)

```
0    P(415).
1    P(185).
1    P(295).
1    P(413).
2    P(34).
2    P(75).
2    P(17).
2    P(170).
2    P(102).
2    P(412).
3    P(45).
3    P(66).
3    P(81).
3    P(100).
3    P(89).
4    P(14).
4    P(23).
4    P(11).
4    P(29).
4    P(48).
```

Multiple proof sketches can be merged by BFS level. This is especially meaningful for multiple proofs of the same theorem but with different extra assumptions.

Other Criteria for Selection / Prioritization

- Domain knowledge (user expertise; collaboration)
- Most recent history (the last proof is likely to be closer to the desired proof)
- Frequency counts (clauses that occur in many proofs are presumably important!)
- Classification methods (ENIGMA, ProofWatch)

Highly recommended (I only learned about it yesterday):

ProofWatch:
Watchlist Guidance for Large Theories in E

Zarathustra Goertzel[1], Jan Jakubův[1], Stephan Schulz[2], and Josef Urban[1⋆]

[1] Czech Technical University in Prague
[2] DHBW Stuttgart

AIM Hints Library

Note: the numbers below refer only to “interesting” proofs, that is, proofs of special cases of interest to loop theorists or proofs of lemmas that look interesting.

- **Before hint prioritization**: 549 proofs in 117 output files, 167K distinct hints, 47K appearing in more than one output file
- **As of November 2018**: 641 proofs in 149 output files, 2.3 million distinct hints, 90K appearing in more than two output files
- **As of January 2019**: 660 proofs in 158 output files, 2.6 million distinct hint clauses, 114K appearing in more than two output files

Experiments: Eliminating Extra Assumptions

- Single goal, 33 extra assumptions to eliminate
- Starting with a proof with all 33 extra assumptions and a library of AIM hints
- Run under different strategies for updating hints after each new proof
- Evaluate by total CPU time spent and total givens to get the proof with no extra assumptions

The Variety and the Goal

The particular variety is of (mild) interest to loop theorists because it includes some well known ones as special cases:

```
(1 / x) * (x * y) = y # label("LIP").
(x * y) * (y \ 1) = x # label("RIP").
(x * y) * (z * (x * y)) = ((x * (y * z)) * x) * y.
((y * x) * z) * (y * x) = y * (x * ((z * y) * x)).
```

The goal is one of the 7 AIM goals:

```
K(a(x,y,z),u) = 1 # label("Ka").
```

Specifics

When hints are first input into PROVER9, they are given their own index numbers. Besides selecting hint matchers by age or weight or whatever, it is also possible to select them by **hint age**, which corresponds to how they are indexed.

All the versions of prioritization in these experiments involve varying the order in which hints are listed and then selecting them by hint age.

In all cases, the next prioritized set of hints in a run consists only of hints from the most recent proof.

Specifics II

PROVER9's basic proof output is in *unexpanded* form, which means that while rewrites are listed (as secondary inferences), the intermediate demodulants (rewritten forms) are not.

There is also an *expanded* form which shows all the intermediate demodulants.

Either form can be converted to hints for use in a hints list.

- Advantage of unexpanded hints: keeps down the number of hints.
- Advantage of expanded hints: sometimes intermediate demodulants are matched directly

Results: Proof Step Order

In these three cases, the next prioritized set consists only of hints from the most recent proof.

- Unexpanded proof in proof step order
  - Total givens: Grand Total = 219242.00
  - Total User CPU time: Grand Total = 1099313.03
- Unexpanded proof in reverse proof step order
  - Total givens: Grand Total = 179602.00
  - Total User CPU time: Grand Total = 236565.12
- Expanded proof in reverse proof step order
  - Total givens: Grand Total = 155384.00
  - Total User CPU time: Grand Total = 140203.96

Results: Variations on BFS

- BFS on unexpanded proof
  - Total givens: Grand Total = 103748.00
  - Total User CPU time: Grand Total = 323327.14
- BFS as above, with intermediate demodulants appended in clause number order
  - Total givens: Grand Total = 77571.00
  - Total User CPU time: Grand Total = 88237.82
- BFS as above, with intermediate demodulants appended in reverse clause number order
  - Total givens: Grand Total = 71555.00
  - Total User CPU time: Grand Total = 33408.87

The Future

It seems to be common in computer science to end talks with "Future Work". Mathematicians usually don't do this unless they are graduate students. Anyway, here we go:

- More dynamic methods (ProofWatch, learning, etc.)
- Reboot the whole AIM project in E just to see what happens. Since E's watchlist has recently become a lot more sophisticated, reproducing AIM results should now be feasible. And maybe we'll find some new ones.
- Settle the AIM Conjecture (a pipe dream?)
- Experiment with other large-scale open problems from quasigroup/loop theory (e.g. the Osborn Problem)