universität innsbruck

mædmax at School

Learning Selection in Equational Reasoning

Sarah Winkler

University of Innsbruck

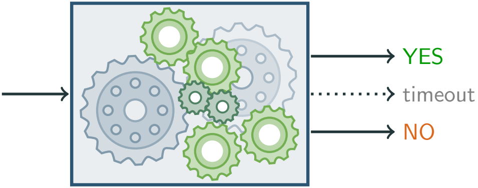
4th Conference on Artificial Intelligence and Theorem Proving

10 April 2019

Equational Theorem Proving

$$0 + x \approx x$$
$$(-x) + x \approx 0$$
$$x + (y + z) \approx (x + y) + z$$
$$x + y \approx y + x$$
$$x \cdot (y \cdot z) \approx (x \cdot y) \cdot z$$
$$(x + y) \cdot z \approx (x \cdot z) + (y \cdot z)$$

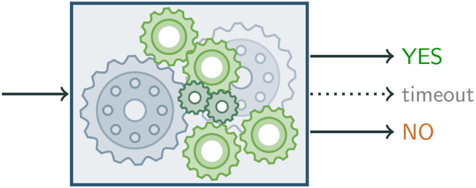
$$\mathsf{a} \cdot 0 \approx 0$$



- input: set of equations $\mathcal{E}_0$ and goal $s \approx t$
- output: YES if $\mathcal{E}_0 \vDash s \approx t$ or NO otherwise

Equational Theorem Proving

$$
\begin{aligned}
0 + x &\approx x \\
(-x) + x &\approx 0 \\
x + (y + z) &\approx (x + y) + z \\
x + y &\approx y + x \\
x \cdot (y \cdot z) &\approx (x \cdot y) \cdot z \\
(x + y) \cdot z &\approx (x \cdot z) + (y \cdot z) \\
\mathsf{a} \cdot 0 &\approx 0
\end{aligned}
$$



- input: set of equations $\mathcal{E}_0$ and goal $s \approx t$
- output: YES if $\mathcal{E}_0 \vDash s \approx t$ or NO otherwise

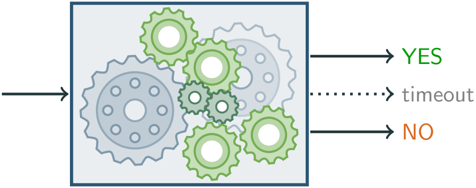
maedmax

equational theorem proving tool based on maximal ordered completion

Equational Theorem Proving

$$
\begin{aligned}
0 + x &\approx x \\
(-x) + x &\approx 0 \\
x + (y + z) &\approx (x + y) + z \\
x + y &\approx y + x \\
x \cdot (y \cdot z) &\approx (x \cdot y) \cdot z \\
(x + y) \cdot z &\approx (x \cdot z) + (y \cdot z)
\end{aligned}
$$

$$\mathsf{a} \cdot 0 \approx 0$$

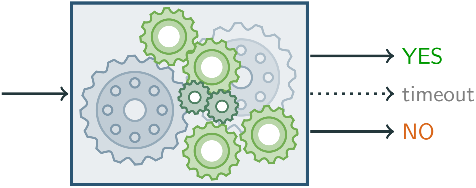


- input: set of equations $\mathcal{E}_0$ and goal $s \approx t$
- output: YES if $\mathcal{E}_0 \vDash s \approx t$ or NO otherwise

maedmax

equational theorem proving tool based on maximal ordered completion
(**ma**edmax: **e**quational **d**eduction **max**imized)

$$
\begin{aligned}
0 + x &\approx x \\
(-x) + x &\approx 0 \\
x + (y + z) &\approx (x + y) + z \\
x + y &\approx y + x \\
x \cdot (y \cdot z) &\approx (x \cdot y) \cdot z \\
(x + y) \cdot z &\approx (x \cdot z) + (y \cdot z) \\
\mathrm{a} \cdot 0 &\approx 0
\end{aligned}
$$



- input: set of equations $\mathcal{E}_0$ and goal $s \approx t$
- output: YES if $\mathcal{E}_0 \vDash s \approx t$ or NO otherwise

maedmax

equational theorem proving tool based on maximal ordered completion (**ma**edmax: **e**quational **d**eduction **max**imized)

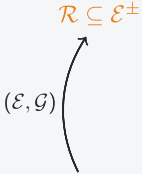
S. Winkler and G. Moser.
Maedmax: A Maximal Ordered Completion Tool.
In *Proc. 9th IJCAR*, LNCS 10900, pp. 472-480, 2018.

Content

Maximal Ordered Completion

0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

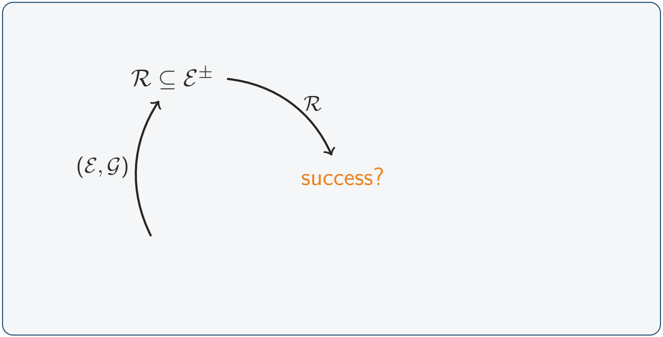
Maximal Ordered Completion



0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

Maximal Ordered Completion

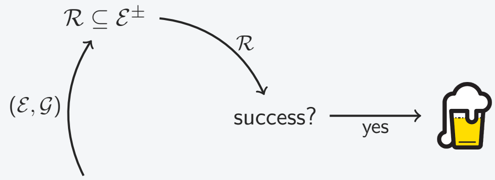


0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

2 check for joinable goal or saturation

Maximal Ordered Completion

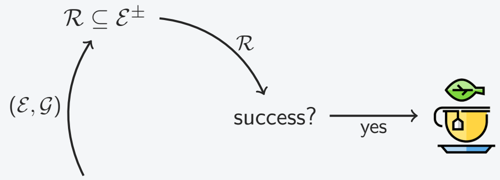


0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

2 check for joinable goal or saturation

Maximal Ordered Completion

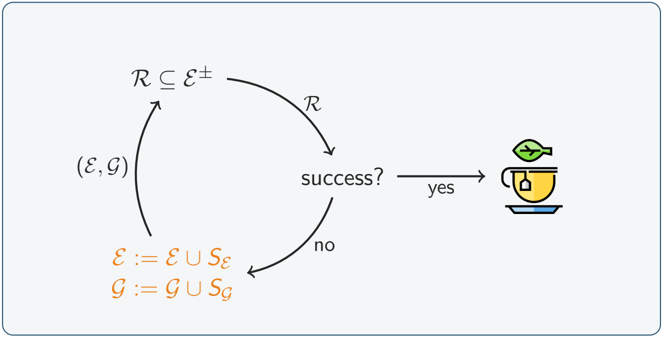


0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

2 check for joinable goal or saturation

Maximal Ordered Completion



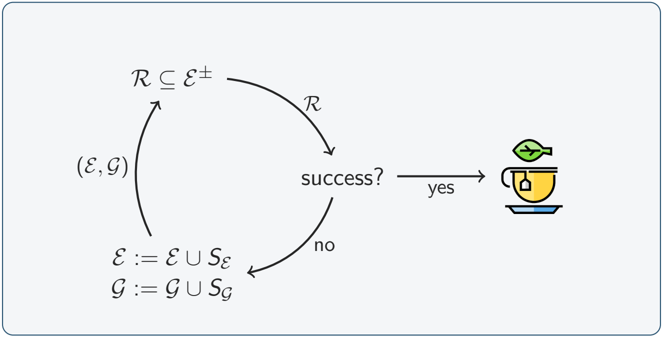
0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

2 check for joinable goal or saturation

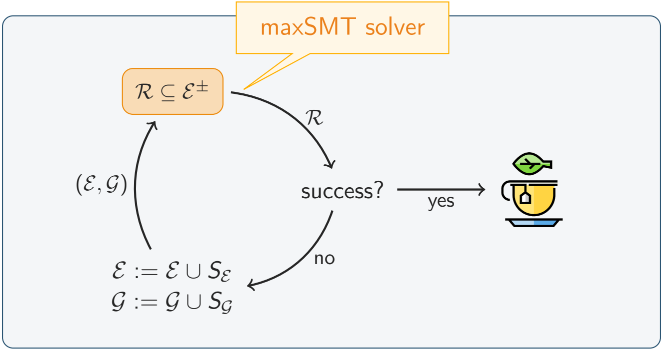
3 add some critical pairs $S_{\mathcal{E}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E})$ and $S_{\mathcal{G}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E}, \mathcal{G})$

Maximal Ordered Completion



0. initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$
1. get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$
2. check for joinable goal or saturation
3. add some critical pairs $S_{\mathcal{E}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E})$ and $S_{\mathcal{G}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E}, \mathcal{G})$, repeat from 1

Maximal Ordered Completion



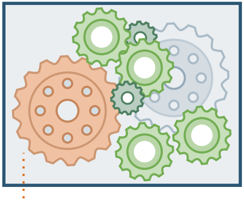
0 initialize equations and goals $(\mathcal{E}, \mathcal{G})$ to $(\mathcal{E}_0,\ \{s \approx t\})$

1 get terminating rewrite system $\mathcal{R}$ from $\mathcal{E}$

2 check for joinable goal or saturation

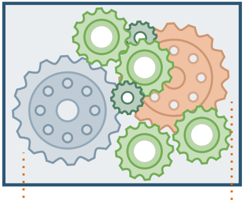
3 add some critical pairs $S_{\mathcal{E}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E})$ and $S_{\mathcal{G}} \subseteq \mathrm{CP}_{>}(\mathcal{R} \cup \mathcal{E}, \mathcal{G})$, repeat from 1

Critical Choice Points



finding $\mathcal{R}$

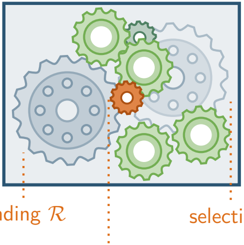
Critical Choice Points



Selecting things

- selection of $S_{\mathcal{E}}$ and $S_{\mathcal{G}}$ is highly critical
- on average only 15% of selected equations and goals used for proof
- learn better choice criteria

Critical Choice Points



Estimating proof progress

- heuristic proof progress estimate: if stuck
  - add additional (old) equations
  - or ultimately restart
- learn better estimate

it's all about the next (small) thing

Content

Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$

Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

N. Augsten, M. Böhlen, and J. Gamper.
The *pq*-gram distance between ordered labeled trees.
ACM Transactions on Database Systems, 35(1):1–36, 2010.

Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$

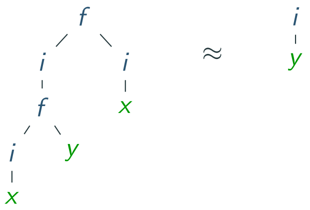
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



Learning Equation Selection

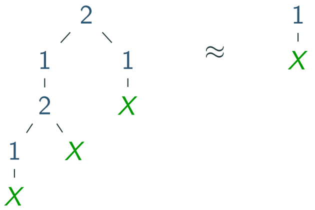
Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$

- abstract symbol names to arities, variables to X



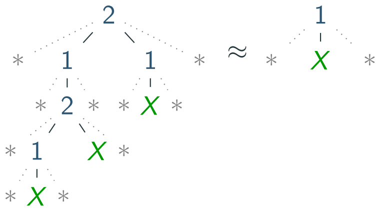
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: ∼200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes

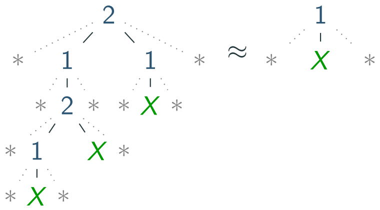
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: "go $p-1$ down, q right"

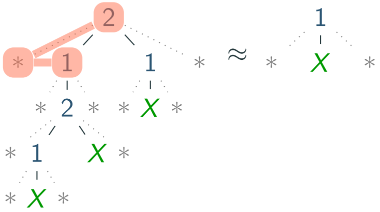
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: "go $p-1$ down, q right"
- for $p = 2$, $q = 1$ obtain
  2.*.1

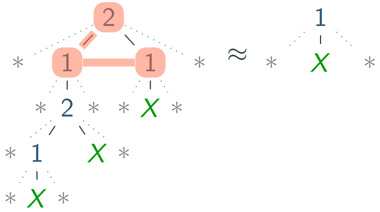
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: “go $p-1$ down, q right”
- for $p=2$, $q=1$ obtain
  2.*.1, 2.1.1

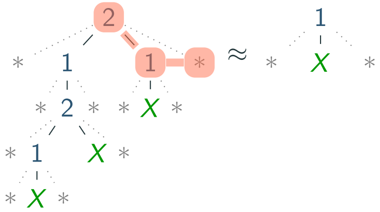
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: ∼200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: "go $p-1$ down, q right"
- for $p = 2$, $q = 1$ obtain
  2.*.1, 2.1.1, 2.1.*

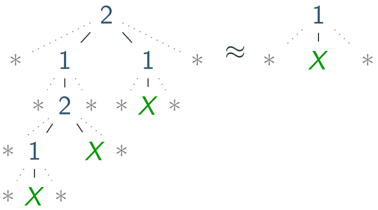
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: "go $p-1$ down, q right"
- for $p = 2$, $q = 1$ obtain
  2.*.1, 2.1.1, 2.1.*, 1.*.2, 1.2.*,
  2.*.1, 2.1.X, 2.X.*, 1.*.X, 1.X.*,
  1.*.X, 1.X.* and 1.*.X, 1.X.*

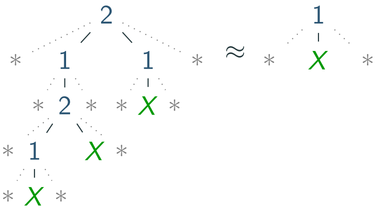
Learning Equation Selection

Features

- state: number of iterations, equations, and goals
- equation:
  - hand-crafted: polarity, size, size difference, age, orientability, linearity, duplicatingness, # of matches and critical pairs on $\mathcal{E}$
  - term structure: $\sim$200 *pq*-gram counts

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract symbol names to arities, variables to X
- add dummy nodes
- *pq*-grams: "go $p - 1$ down, q right"
- for $p = 2$, $q = 1$ obtain
  2.*.1, 2.1.1, 2.1.*, 1.*.2, 1.2.*,
  2.*.1, 2.1.X, 2.X.*, 1.*.X, 1.X.*,
  1.*.X, 1.X.* and 1.*.X, 1.X.*
- count up to arity 3

Learning Equation Selection

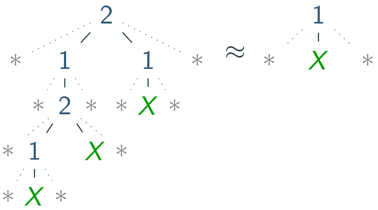
Features

- state: number of iterations, equations, and go
- equation:
  - hand-crafted: polarity, size, size differen
    linearity, duplicatingness, # of matches
  - term structure: ∼200 *pq*-gram counts

1	polarity	0
2	size	10
3	size diff	6
4	age	42
	⋮	
16	2.2.1	0
17	2.*.1	2
18	2.1.1	1
19	2.1.0	0
20	1.*.X	2
21	1.X.*	2
	⋮	
121	2.2.1	0
122	2.*.1	0
123	2.1.1	0
124	2.1.0	0
125	1.*.X	1
126	1.X.*	1
	⋮	

Example (*pq*-Grams)

$\mathsf{f}(\mathsf{i}(\mathsf{f}(\mathsf{i}(x), y)), \mathsf{i}(x)) \approx \mathsf{i}(y)$



- abstract sy
  variables to
- add dummy
- *pq*-grams: ht"
- for $p = 2$,
  2.*.1, 2.1.1
  2.*.1, 2.1.X
  1.*.X, 1.X.
- count up to

Binary classification

classify equation as positive (useful) if occurs in proof, negative otherwise

Binary classification

classify equation as positive (useful) if occurs in proof, negative otherwise

Data collection

- run mædmax in different strategies, recording selections
- check classification of all selected literals upon successful proof
- only 15% positives: duplicate positives for balancing

Binary classification

classify equation as positive (useful) if occurs in proof, negative otherwise

Data collection

- run mædmax in different strategies, recording selections
- check classification of all selected literals upon successful proof
- only 15% positives: duplicate positives for balancing

Classifier

random forest of 100 trees and maximal depth 14

- fast evaluation, in tests slightly better recall than SVC, extra trees

Binary classification

classify equation as positive (useful) if occurs in proof, negative otherwise

Data collection

- run mædmax in different strategies, recording selections
- check classification of all selected literals upon successful proof
- only 15% positives: duplicate positives for balancing

Classifier

random forest of 100 trees and maximal depth 14

- fast evaluation, in tests slightly better recall than SVC, extra trees

Setup

- evaluate data with Python Scikit
- export random forest to json, load into mædmax (OCaml)

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$

Results

	solved	useful
$\mathcal{M}_0$	206	8%

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$
$\mathcal{M}_0$	206	8%	$114K$

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%				

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$			

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$
- ...

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$	0.77	0.83	0.8
$\mathcal{M}_2$	423	15%	$528K$	0.76	0.83	0.79
$\mathcal{M}_3$	433	14%	$704K$	0.78	0.8	0.79

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$
- ...

Results

time spent on selection rises from 0% (random) to 8-10%

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$	0.77	0.83	0.8
$\mathcal{M}_2$	423	15%	$528K$	0.76	0.83	0.79
$\mathcal{M}_3$	433	14%	$704K$	0.78	0.8	0.79

- run on 897 UEQ problems in TPTP, 60s timeout

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$
- ...

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$	0.77	0.83	0.8
$\mathcal{M}_2$	423	15%	$528K$	0.76	0.83	0.79
$\mathcal{M}_3$	433	14%	$704K$	0.78	0.8	0.79

- run on 897 UEQ problems in TPTP, 60s timeout
- other baselines: size solves 575, fifo 366, best 609

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$
- ...

combination with old strategy solves 625

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$	0.77	0.83	0.8
$\mathcal{M}_2$	423	15%	$528K$	0.76	0.83	0.79
$\mathcal{M}_3$	433	14%	$704K$	0.78	0.8	0.79

- run on 897 UEQ problems in TPTP, 60s timeout
- other baselines: size solves 575, fifo 366, best 609

Reinforcement loop experiment

- start with random selection $\mathcal{M}_0$
- collect selections $\mathcal{S}_0$ running $\mathcal{M}_0$ with 3 different strategies
- build classifier: $\mathcal{M}_1$ selects literals e with $P_{\text{positive}}(e) > 0.4$
- selections $\mathcal{S}_1$: run $\mathcal{M}_1$ with 3 different strategies, add to $\mathcal{S}_0$
- ...

Results

	solved	useful	$\lvert\mathcal{S}_i\rvert$	precision	recall	f1
$\mathcal{M}_0$	206	8%	$114K$	0.86	0.94	0.9
$\mathcal{M}_1$	409	14%	$358K$	0.77	0.83	0.8
$\mathcal{M}_2$	423	15%	$528K$	0.76	0.83	0.79
$\mathcal{M}_3$	433	14%	$704K$	0.78	0.8	0.79

- run on 897 UEQ problems in TPTP, 60s timeout
- other baselines: size solves 575, fifo 366, best 609
- feature importance: 60% *pq*-grams, 40% hand crafted
  most useful: literal size, # active literals, # matches and CPs

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- \# SMT checks
- \# CPs with $\mathcal{R}$
- cost of last maxSMT check
- \# facts in $\mathcal{E}$ reducible by last $\mathcal{R}$
- ...

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- ...
- # SMT checks
- # CPs with $\mathcal{R}$
- cost of last maxSMT check
- # facts in $\mathcal{E}$ reducible by last $\mathcal{R}$

Data collection

- collect TSTP proofs from mædmax, E, Vampire

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- ...
- # SMT checks
- # CPs with $\mathcal{R}$
- cost of last maxSMT check
- # facts in $\mathcal{E}$ reducible by last $\mathcal{R}$

Data collection

- collect TSTP proofs from mædmax, E, Vampire
- add proof tracking mode to mædmax:
  given input proof P, check progress wrt P in every iteration

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- ...
- # SMT checks
- # CPs with $\mathcal{R}$
- cost of last maxSMT check
- # facts in $\mathcal{E}$ reducible by last $\mathcal{R}$

Data collection

- collect TSTP proofs from mædmax, E, Vampire
- add proof tracking mode to mædmax:
  given input proof P, check progress wrt P in every iteration
  (progress: unseen literals to passive set, passive literals to active set)

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- ...
- # SMT checks
- # CPs with $\mathcal{R}$
- cost of last maxSMT check
- # facts in $\mathcal{E}$ reducible by last $\mathcal{R}$

Data collection

- collect TSTP proofs from mædmax, E, Vampire
- add proof tracking mode to mædmax:
  given input proof P, check progress wrt P in every iteration
  (progress: unseen literals to passive set, passive literals to active set)
- store difference of proof state vectors between two iterations, along with progress classification

Learning Proof Progress

State features

- size of $\mathcal{E}$
- memory used
- ...
- # SMT checks
- # CPs with $\mathcal{R}$
- cost of last maxSMT check
- # facts in $\mathcal{E}$ reducible by last $\mathcal{R}$

Data collection

- collect TSTP proofs from mædmax, E, Vampire
- add proof tracking mode to mædmax:
  given input proof P, check progress wrt P in every iteration
  (progress: unseen literals to passive set, passive literals to active set)
- store difference of proof state vectors between two iterations, along with progress classification
- get data of about 20K iterations

Classifier

- random forest of 100 trees and maximal depth 10
- binary classification: progress or no progress

Classifier

- random forest of 100 trees and maximal depth 10
- binary classification: progress or no progress

Evaluation

- cross-validated precision and recall of 0.72
- manually design new decision tree based on most influential features: gain 1.5% new problems with best strategy

Classifier

- random forest of 100 trees and maximal depth 10
- binary classification: progress or no progress

Evaluation

- cross-validated precision and recall of 0.72
- manually design new decision tree based on most influential features: gain 1.5% new problems with best strategy

Overall gain

+4.5% solved problems combining new selection classifier with old selection, and adding new progress estimate

What worked

- a bit more 

What worked

- a bit more 

What did not work

- taking symbol names into account: TPTP problems too diverse?

What worked

- a bit more 

What did not work

- taking symbol names into account: TPTP problems too diverse?

What's next

- more data
- more/other features
  - more state features?
  - longer *pq*-grams, or vertical ENIGMA features?
- more experiments
  - how to combine with previous selection strategy?
  - use proof progress estimate also for restarts?
- ...