Automation of Induction in Saturation

Petra Hozzová and Laura Kovács

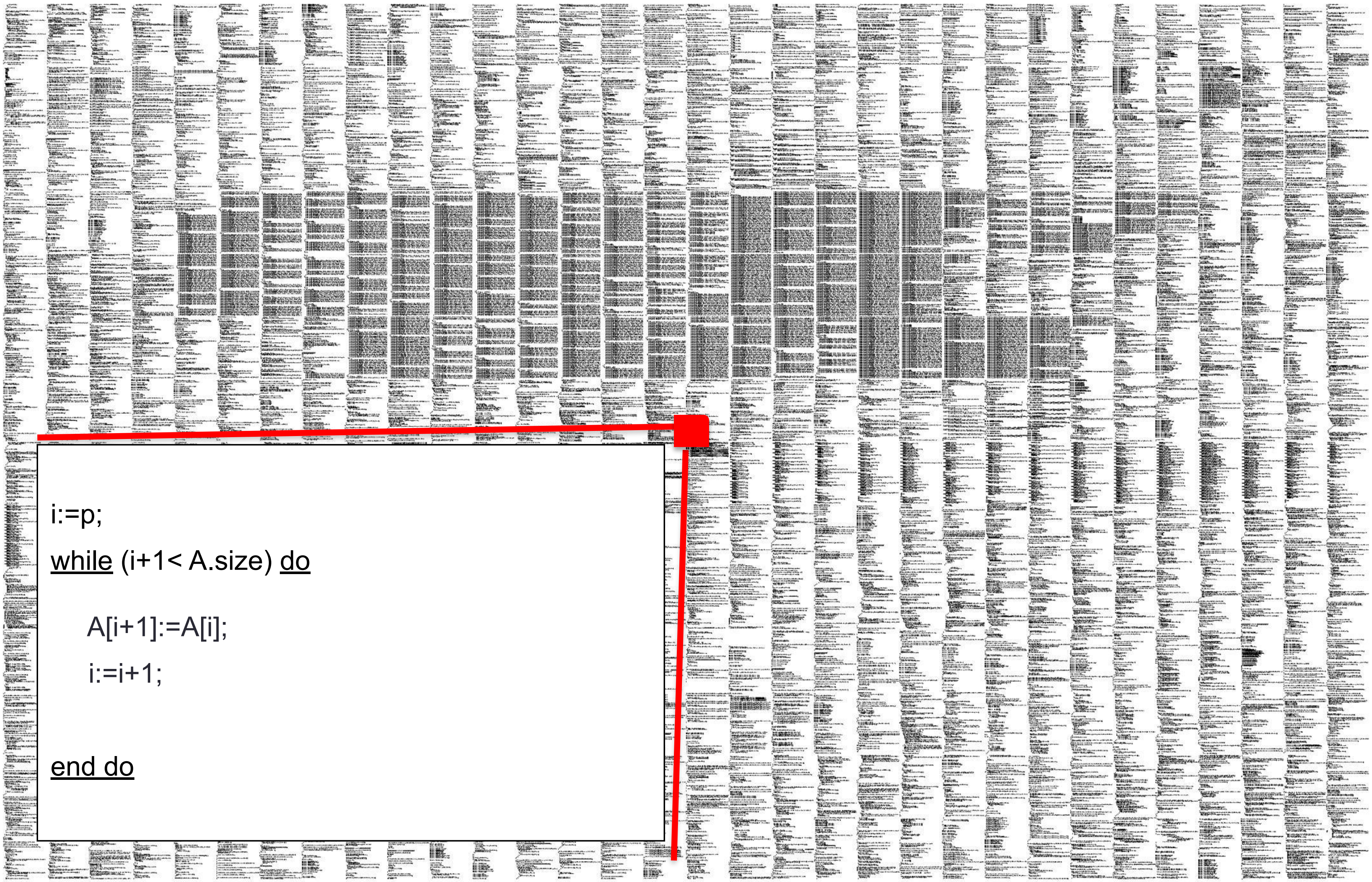
for(syte) TU WIEN Informatics

Automation of Induction for Program Analysis

Automation of Induction for Program Analysis

(ex. ~250kLoC, Vampire prover)

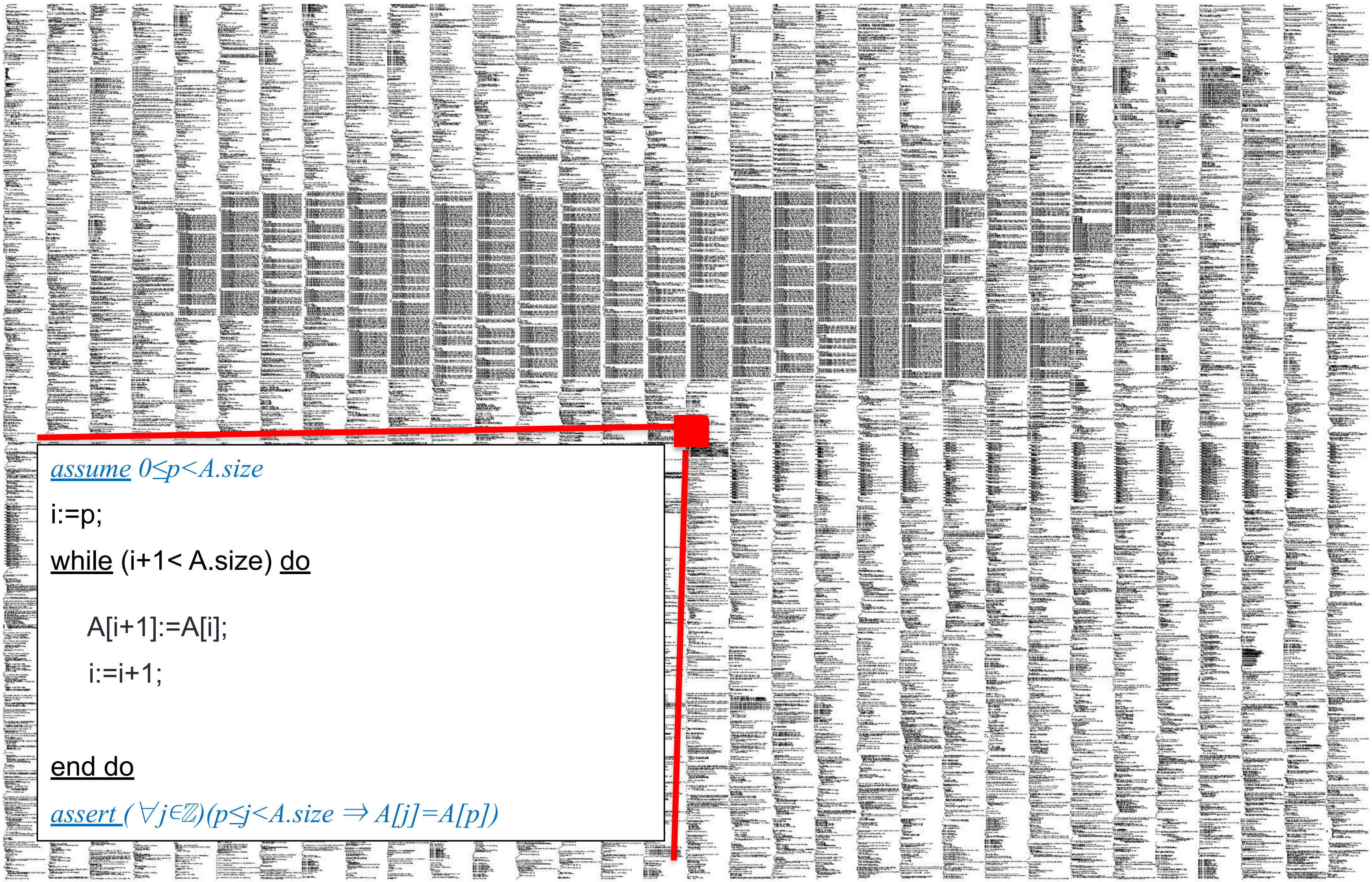
Automation of Induction for Program Analysis



```
i:=p;
while (i+1< A.size) do
    A[i+1]:=A[i];
    i:=i+1;

end do
```

Automation of Induction for Program Analysis



```
assume 0≤p<A.size
i:=p;
while (i+1< A.size) do
    A[i+1]:=A[i];
    i:=i+1;
end do
assert (∀j∈ℤ)(p≤j<A.size ⇒ A[j]=A[p])
```

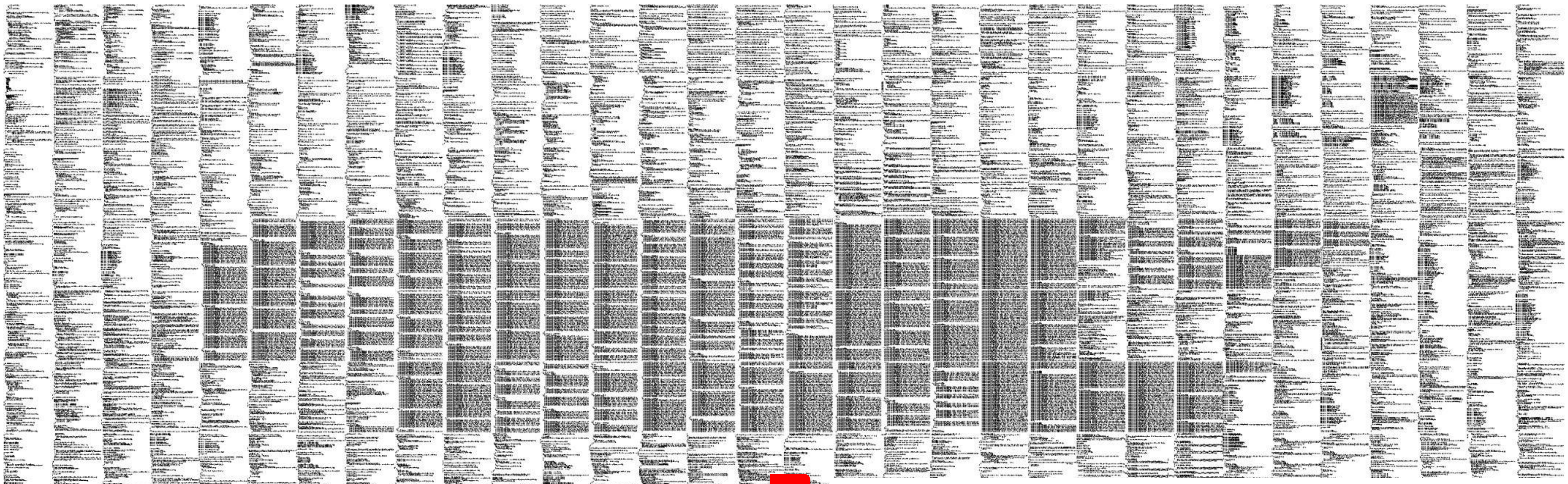
Automation of Induction for Program Analysis

```
assume 0≤p<A.size
i:=p;
while (i+1< A.size) do

    A[i+1]:=A[i];
    i:=i+1;
    invariant (∀j∈ℤ)(p≤j<i ⇒ A[j+1]=A[j])
end do
assert (∀j∈ℤ)(p≤j<A.size ⇒ A[j]=A[p])
```

1. Generating inductive loop properties
2. Proving inductive properties over integers

Automation of Induction for Program Analysis



assume $0 \leq p < A.size$

i:=p;

while (i+1< A.size) do

    A[i+1]:=A[i];

    i:=i+1;

    invariant $(\forall j \in \mathbb{Z})(p \leq j < i \Rightarrow A[j+1]=A[j])$

end do

assert $(\forall j \in \mathbb{Z})(p \leq j < A.size \Rightarrow A[j]=A[p])$

1. Generating inductive loop properties
2. Proving inductive properties over integers

by saturation-based first-order theorem proving

Automation of Induction for Program Analysis

```
dataype nat = zero | succ of nat

add zero x = x
add (succ x) y = succ (add x y)
```

Automation of Induction for Program Analysis

<u>dataype</u> nat = zero | succ <u>of</u> nat

add zero x = x

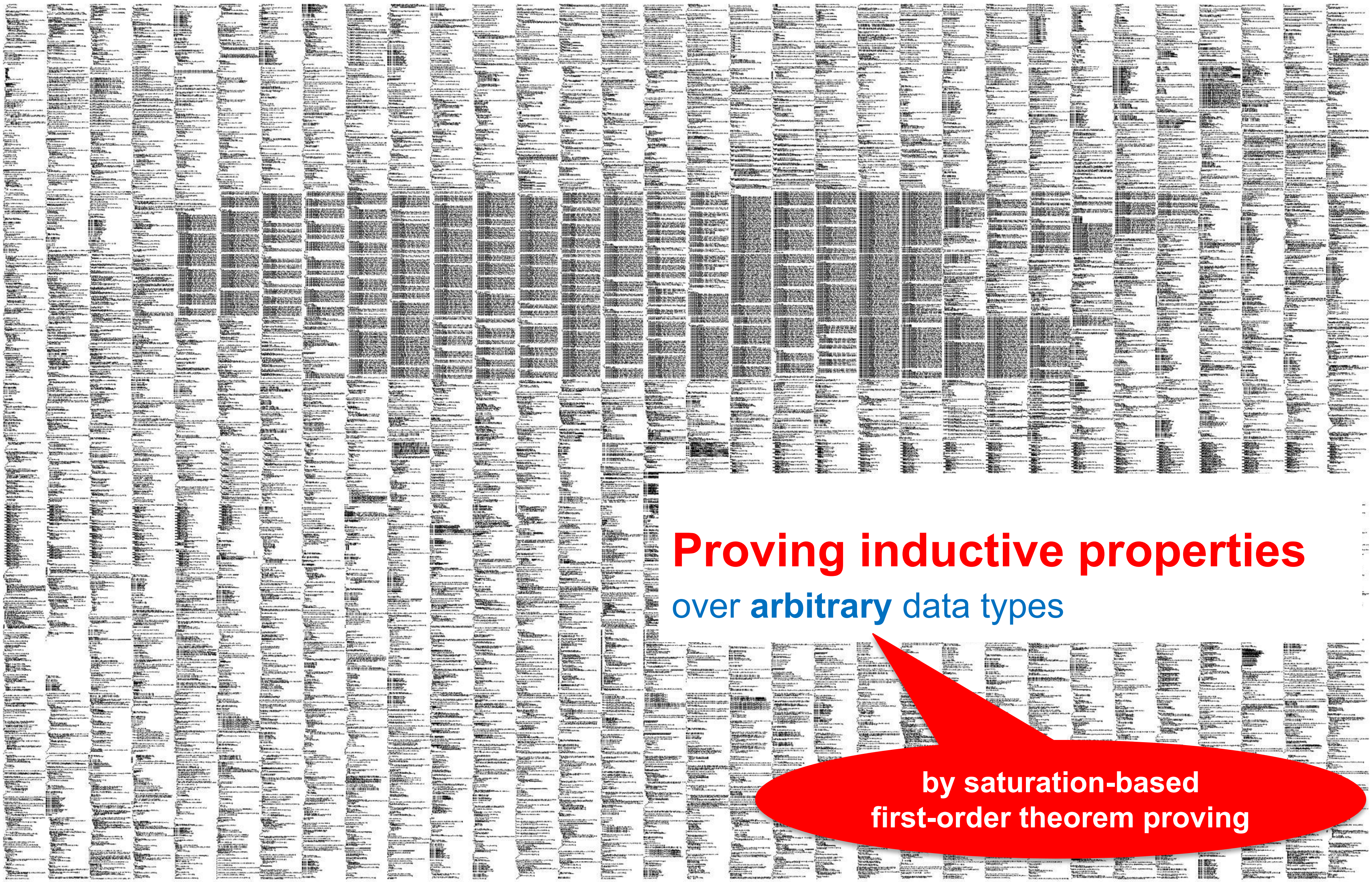
add (succ x) y = succ (add x y)

<u>assert</u> $(\forall x)\ (\forall y)\ (add\ x\ y) = (add\ y\ x)$

Proving inductive properties over naturals

by saturation-based
first-order theorem proving

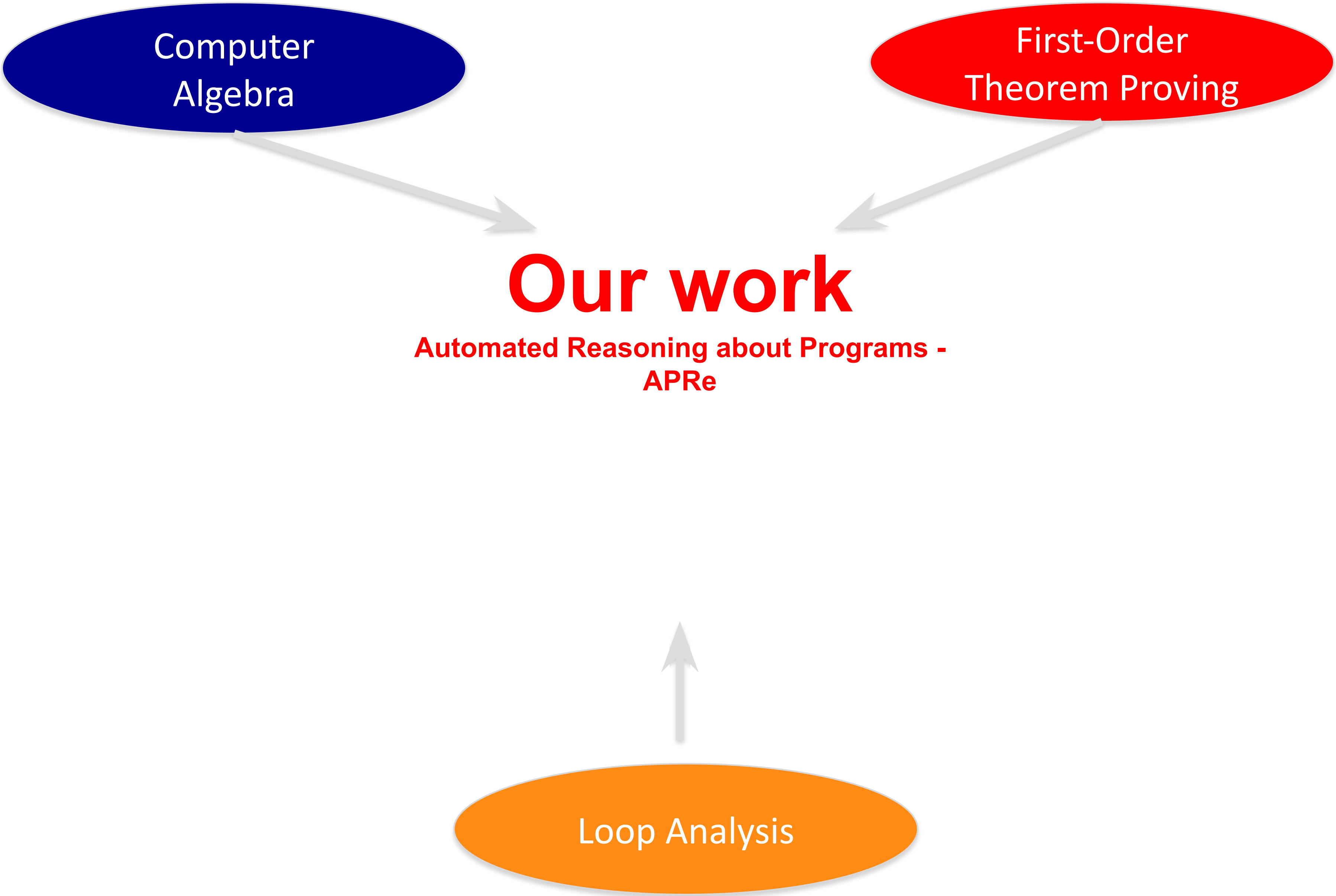
Automation of Induction for Program Analysis



Proving inductive properties

over **arbitrary** data types

by saturation-based first-order theorem proving


Computer Algebra
First-Order Theorem Proving
Our work
Automated Reasoning about Programs - APRe
Loop Analysis

Computer Algebra

First-Order Theorem Proving

Our work

Automated Reasoning about Programs - APRe

FWF Der Wissenschaftsfonds.

erc European Research Council Supporting top researchers from anywhere in the world

WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

amazon webservices™

Loop Analysis

Automation of Induction

in

Saturation-Based Proof Search

Automation of Induction

in

Saturation-Based Proof Search

Joint work with Márton Hajdu, Giles Reger, Andrei Voronkov

1. Saturation in first-order theorem proving
2. Saturation with induction over term algebras
3. Saturation with induction over integers

Automation of Induction

in

Saturation-Based Proof Search

1. Saturation in first-order theorem proving
2. Saturation with induction over term algebras
3. Saturation with induction over integers

Part 1 | What Can We Prove?

Formulas in first-order logic* with quantifiers and theories, such as:

- group idempotency ⇒ commutativity
- $1 + 2 + \ldots + n = n*(n+1)/2$
- verification problems
- …

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition 12,11]
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Termination reason: Refutation
```

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition 12,11]
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Termination reason: Refutation
```

proof by refutation

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Termination reason: Refutation
```

inferences deriving new formulas from 0+ other formulas

proof by refutation

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Termination reason: Refutation
```

preprocessing

inferences deriving new formulas from 0+ other formulas

proof by refutation

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Termination reason: Refutation
```

preprocessing

inferences deriving new formulas from 0+ other formulas

superposition calculus

proof by refutation

Part 1 | Example Proof

```
1. ! [X0] : mult(e,X0) = X0 [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
4. ! [X0] : e = mult(X0,X0) [input]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
8. ? [X0,X1] : mult(X0,X1)!=mult(X1,X0) => mult(sK0,sK1)!=mult(sK1,sK0) [choice axiom]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
10. mult(e,X0) = X0 [cnf transformation 1]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
12. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
13. e = mult(X0,X0) [cnf transformation 4]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
15. mult(X0,mult(X0,X1)) = mult(e,X1) [superposition 12,13]
17. mult(inverse(X4),mult(X4,X5)) = mult(e,X5) [superposition
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
75. mult(X4,mult(X3,X4)) = mult(inverse(X3),e) [superposition 22,19]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
269. mult(sK0,sK1) != mult(sK0,sK1) [superposition 14,125]
270. $false [trivial inequality removal 269]
…
% Terminat   efutation
```

preprocessing

inferences deriving new formulas from 0+ other formulas

superposition calculus

unused formulas

proof by refutation

Part 1 | Saturation-Based Theorem Proving

Proving formula F w.r.t. a set of formulas S:

1. Negate F and add $\neg F$ to S
2. Repeat:
   2.1 Choose $G \in S$
   2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   2.3 Add $C_1, \ldots, C_n$ to S
   2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

 Saturation-Based Theorem Proving

Proving formula F w.r.t. a set of formulas S:

1. Negate F and add $\neg F$ to S
2. Repeat:
   2.1 Choose $G \in S$
   2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   2.3 Add $C_1, \ldots, C_n$ to S
   2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

G is an arbitrary formula,
not necessarily F

Proving formula F w.r.t. a set of formulas S:

1. Negate F and add ¬F to S
2. Repeat:
   - 2.1 Choose $G \in S$
   - 2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   - 2.3 Add $C_1, \ldots, C_n$ to S
   - 2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

Part 1 | Saturation-Based Theorem Proving in Vampire

Proving formula F w.r.t. a set of formulas S:

1. Negate F and add ¬F to S
2. Repeat:
   - 2.1 Choose $G \in S$
   - 2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   - 2.3 Add $C_1, \ldots, C_n$ to S
   - 2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

G is an arbitrary formula, not necessarily F

A lot of possibilities, using resolution/superposition over clauses

Part 1 | Saturation-Based Theorem Proving in Vampire

Proving formula F w.r.t. a set of formulas S:

1. Negate F and add ¬F to S
2. Repeat:
   - 2.1 Choose $G \in S$
   - 2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   - 2.3 Add $C_1, \ldots, C_n$ to S
   - 2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

G is an arbitrary formula, not necessarily F

A lot of possibilities, using resolution/superposition over clauses

$$\frac{l = r \lor C \qquad L[l] \lor D}{L[r] \lor C \lor D}$$

Automation of Induction

in

Saturation-Based Proof Search

1. Saturation in first-order theorem proving
2. Saturation with induction over term algebras
3. Saturation with induction over integers

Induction can be implemented by reducing goals to subgoals,

so having a goal $\forall x\, F[x]$ you can prove it by induction on x.

Part 2 | Saturation and Induction

Induction can be implemented by reducing goals to subgoals,

so having a goal $\forall x\, F[x]$ you can prove it by induction on x.

But . . .
saturation theorem proving is not about reducing goals to subgoals.

Part 2 | Saturation and Induction

Induction can be implemented by reducing goals to subgoals,

so having a goal $\forall x\, F[x]$ you can prove it by induction on x.

But . . .
saturation theorem proving is not about reducing goals to subgoals.

1. Negate F and add ¬F to S
2. Repeat:
   - 2.1 Choose G ∈ S
   - 2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   - 2.3 Add $C_1, \ldots, C_n$ to S
   - 2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

Part 2 | Saturation and Induction

Induction can be implemented by reducing goals to subgoals,

so having a goal $\forall x\, F[x]$ you can prove it by induction on x.

But . . .
saturation theorem proving is not about reducing goals to subgoals.

1. Negate F and add $\neg F$ to S
2. Repeat:
   - 2.1 Choose $G \in S$
   - 2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S
   - 2.3 Add $C_1, \ldots, C_n$ to S
   - 2.4 If S contains false, return *"F is valid"*
3. Return *"F is not valid"*

resolution/superposition
and induction

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x]\ )}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation and induction

1. Negate F and add ¬F to S
2. Repeat:

2.1 Choose G ∈S $\qquad \neg L[a] \vee C$

2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S

2.3 Add $C_1, \ldots, C_n$ to S $\qquad CNF(premise \rightarrow \forall x L[x])$

2.4 If S contains false, return *"F is valid"*

3. Return *"F is not valid"*

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation and induction

1. Negate F and add ¬F to S
2. Repeat:

2.1 Choose $G \in S$ $\neg L[a] \vee C$

2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S

2.3 Add $C_1, \ldots, C_n$ to S $CNF(premise \rightarrow \forall x L[x])$

2.4 If S contains false, return *"F is valid"*

3. Return *"F is not valid"*

- becomes an ordinary formula in S

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation and induction

1. Negate F and add ¬F to S
2. Repeat:

2.1 Choose G ∈S $\qquad \neg L[a] \vee C$

2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S

2.3 Add $C_1, \ldots, C_n$ to S $\qquad CNF(premise \rightarrow \forall x L[x])$

2.4 If S contains false, return *"F is valid"*

3. Return *"F is not valid"*

- becomes an arbitrary formula in S

- not necessarily used to prove $\neg L[a]$ as a subgoal

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation tailored to induction

– upon applying the induction rule in saturation (step 2.2),

– resolve $CNF(\neg premise \vee \forall x L[x])$ with $\neg L[a] \vee C$,

– add $CNF(\neg premise \vee C)$ to the search space S (step 2.3).

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation tailored to induction: Induction + Resolution

– upon applying the induction rule in saturation (step 2.2),

– resolve $CNF(\neg premise \vee \forall x L[x])$ with $\neg L[a] \vee C$,

– add $CNF(\neg premise \vee C)$ to the search space S (step 2.3).

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation tailored to induction: Induction + Resolution

Repeat:

2.1 Choose $G \in S$ — $\neg L[a] \vee C$

2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S — Induction+ Resolution

2.3 Add $C_1, \ldots, C_n$ to S — $CNF(\neg premise \vee C)$

2.4 If S contains false, return *"F is valid"*

Part 2 | Saturation and Induction over term algebras

- Induction rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall x L[x])}$$

for a valid induction scheme $premise \rightarrow \forall x L[x]$

E.g.: $(L[0] \wedge \forall x(L[x] \rightarrow L[x+1])) \rightarrow \forall x L[x]$

- Saturation tailored to induction: Induction + Resolution

Repeat:

2.1 Choose $G \in S$ — $\neg L[a] \vee C$

2.2 Derive consequences $C_1, \ldots, C_n$ of G and formulas from S — Induction+ Resolution

2.3 Add $C_1, \ldots, C_n$ to S — $CNF(\neg premise \vee C)$

2.4 If S contains false, return <u>*"F is valid"*</u>

In a way, $L[a]$ is handled as a subgoal we try to prove.

Example: Prove commutativity of addition on naturals

$$\forall x \forall y\ (x+y = y+x)$$

where + is the add function, recursively defined

Example: Prove commutativity of addition on naturals

$$\forall x \forall y\ (x+y = y+x)$$

1. Negate and skolemize:

$$\sigma_0+\sigma_1 \neq \sigma_1+\sigma_0$$

where σ_0 are skolem functions.

Part 2 | Saturation and Induction over term algebras

Example: Prove commutativity of addition on naturals

$$\forall x \forall y\ (x+y = y+x)$$

1. Negate and skolemize:

$$\sigma_0+\sigma_1 \neq \sigma_1+\sigma_0$$

where σ_0 are skolem functions.

2.1. Choose and decide that we use induction on σ_0:

$$\sigma_0+\sigma_1 \neq \sigma_1+\sigma_0$$

Example: Prove commutativity of addition on naturals

$$\forall x \forall y\, (x+y = y+x)$$

1. Negate and skolemize:

$$\sigma_0+\sigma_1 \neq \sigma_1+\sigma_0$$

where σ_0 are skolem functions.

2.1. Choose and decide that we use induction on σ_0:

$$\sigma_0+\sigma_1 \neq \sigma_1+\sigma_0$$

2.2. Use the induction schema:

$$(F[0] \wedge \forall x(F[x] \rightarrow F[x+1])) \rightarrow \forall x(F[x])$$

and add the induction axiom/formula, using the induction rule:

$$(\, 0+\sigma_1 = \sigma_1+0 \quad \wedge \quad \forall x(x+\sigma_1 = \sigma_1+x \rightarrow (x+1)+\sigma_1 = \sigma_1+(x+1))\,)$$

$$\rightarrow \forall x\, (x+\sigma_1=\sigma_1+x)$$

2.2. Skolemize the induction axiom and convert it to CNF:

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2 \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1) \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

Part 2 | Saturation and Induction over term algebras

2.2. Skolemize the induction axiom and convert it to CNF:

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2 \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1) \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

2.2. Resolve the resulting clauses against the (negated) goal:

$$\sigma_1+\sigma_0 \neq \sigma_1+\sigma_0$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2 \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1) \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

Part 2 | Saturation and Induction over term algebras

2.2. Skolemize the induction axiom and convert it to CNF:

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2 \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1) \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

2.2. Resolve the resulting clauses against the (negated) goal:

$$\sigma_1+\sigma_0 \neq \sigma_1+\sigma_0$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2 \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1) \quad \vee \quad x+\sigma_1 = \sigma_1+x$$

2.3. Add the such obtained clauses to the search space:

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

Note that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- are stronger than the induction axiom

Note that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \lor \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \lor \quad (\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- are stronger than the induction axiom
- are friendly to saturation theorem provers
  - are quantifier-free
    (so good for SAT/SMT + first-order reasoning)
  - cannot be used in many inferences
    (the only positive equality is $\sigma_2+\sigma_1 = \sigma_1+\sigma_2$ and σ_2 is fresh)

Note that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- are stronger than the induction axiom
- are friendly to saturation theorem provers
  - are quantifier-free
    (so good for SAT/SMT + first-order reasoning)
  - cannot be used in many inferences
    (the only positive equality is $\sigma_2+\sigma_1 = \sigma_1+\sigma_2$ and σ_2 is fresh)

Thus, applying thousands of induction inferences during proof search would hardly affect the prover's performance.

Note also that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- can be targeted by further applications of induction

Note also that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- can be targeted by further applications of induction
- obtain the clauses:

$$\sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$(\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

Note also that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- can be targeted by further applications of induction
- obtain the clauses:

$$\sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$(\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- these clauses are very similar to clauses we would obtain using mathematical induction proofs:
  - assert $\sigma_2+\sigma_1 = \sigma_1+\sigma_2$
  - prove $(\sigma_2+1) + \sigma_1 \neq \sigma_1+(\sigma_2+1)$

Note also that the added clauses

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad \sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$0+\sigma_1 \neq \sigma_1+0 \quad \vee \quad (\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- can be targeted by further applications of induction
- obtain the clauses:

$$\sigma_2+\sigma_1 = \sigma_1+\sigma_2$$

$$(\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$$

- these clauses are very similar to clauses we would obtain using mathematical induction proofs:
    - assert $\sigma_2+\sigma_1 = \sigma_1+\sigma_2$
    - prove $(\sigma_2+1)+\sigma_1 \neq \sigma_1+(\sigma_2+1)$

But, we perform all these steps within the saturation framework!

Summary

- Induction rule over term algebras

$$\frac{\neg L[a] \lor C}{CNF(premise \rightarrow \forall x L[x])}$$

  for a valid induction scheme $premise \rightarrow \forall x L[x]$

- Saturation tailored to induction: Induction + Resolution

Summary

- Induction rule over term algebras

$$\frac{\neg L[a] \lor C}{CNF(premise \rightarrow \forall \mathrm{x} \mathrm{L}[\mathrm{x}])}$$

  for a valid induction scheme $premise \rightarrow \forall \mathrm{x}\mathrm{L}[\mathrm{x}]$

- Saturation tailored to induction: Induction + Resolution

Genericity

- Structural induction on any term algebra (e.g. lists, trees)

- Any valid induction scheme can be used

Natural Extension: Induction with Generalization

- Proving stronger/generalized formulas

Natural Extension: Induction with Generalization

- Proving stronger/generalized formulas

- To prove $\forall x\ (x+(x+x) = (x+x)+x)$, use an induction axiom for

  $\forall x \forall y\ (y+(x+x) = (y+x)+x)$ or $\forall x \forall y\ (x+y = y+x)$

  in addition to an axiom for $\forall x\ (x+(x+x) = (x+x)+x)$

Natural Extension: Induction with Generalization

- Induction with generalization rule over term algebras

$$\frac{\neg\, L[a] \vee C}{CNF(premise \rightarrow \forall y L'[y])}$$

where L' is a generalization of L and $premise \rightarrow \forall y L'[y]$ is a valid induction scheme

Natural Extension: Induction with Generalization

- Induction with generalization rule over term algebras

$$\frac{\neg L[a] \vee C}{CNF(premise \rightarrow \forall y L'[y]\ )}$$

where L' is a generalization of L and $premise \rightarrow \forall y L'[y]$ is a valid induction scheme

- We do not replace L by L' in the search space
- During saturation, both induction rules are used (w and w/o generalization)
- We add induction axioms for both L and L' to the search space

Part 2 | Saturation and Induction over term algebras

	Theory	Vampire*	Vampire	Cvc4	Zipperposition	Zeno	Imandra	Acl2	Cvc4-Gen	ZipRewrite
$\forall x, y.(x + y = y + x)$	$\mathbb{N}$	✓	✓	✓	✓	✓	✓	–	✓	✓
$\forall x.(x + s(x) = s(x + x))$		✓	–	–	–	–	–	–	✓	✓
$\forall x, y, z.(x + (y + z) = (x + y) + z)$		✓	✓	✓	✓	✓	✓	✓	✓	✓
$\forall x.(x + (x + x) = (x + x) + x)$		✓	–	–	–	✓	–	–	✓	✓
$\forall x.((x+x)+((x+x)+x) = x+(x+((x+x)+x)))$		✓	–	–	–	✓	–	–	✓	✓
$\forall x, y.(y + (x + x) = (x + y) + x)$		✓	–	–	–	–	–	–	–	✓
$\forall x.(x \leq x)$		✓	✓	✓	✓	✓	✓	✓	✓	✓
$\forall x, y.(x \leq x + y)$		✓	✓	✓	✓	✓	✓	✓	✓	✓
$\forall x.(x \leq x + x)$		✓	–	–	–	–	–	–	–	–
$\forall x.(x + x \leq (x + x) + x)$		✓	–	–	–	✓	–	–	–	–
$\forall l, k, j.(l +\!\!+ (k +\!\!+ j) = (l +\!\!+ k) +\!\!+ j)$	$\mathbb{L}$	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\forall l.(l +\!\!+ (l +\!\!+ l) = (l +\!\!+ l) +\!\!+ l)$		✓	–	–	–	–	–	–	–	✓
$\forall l, k.(l +\!\!+ (k +\!\!+ (l +\!\!+ l)) = (l +\!\!+ k) +\!\!+ (l +\!\!+ l))$		✓	–	–	–	–	–	–	–	✓
$\forall l, k.\mathtt{prefix}(l, l +\!\!+ k)$		✓	✓	✓	✓	✓	✓	✓	✓	✓
$\forall l.\mathtt{prefix}(l, l +\!\!+ l)$		✓	–	–	–	–	–	–	–	–
$\forall l : \mathbb{L}, x : \mathbb{N}.(\mathtt{cons}(x + s(x), l) +\!\!+ (l +\!\!+ l) = (\mathtt{cons}(s(x) + x, l) +\!\!+ l) +\!\!+ l)$	$\mathbb{N}, \mathbb{L}$	✓	–	–	–	–	–	–	–	–

Implementation Questions

- Which induction schemas to use

- Where to apply induction: clauses, literals, terms

Implementation Questions and Solutions in Vampire

- Which induction schemas to use
  - Induction schemas are implemented individually, controlled by options

- Where to apply induction: clauses, literals, terms
  - Options for selecting literals/terms: negative, uninterpreted constants, uninterpreted constants in the given goal…
  - Options to limit the induction depth
  - Options to limit inductive generalizations
  - Rules and options for multi-clause induction for combining multiple literals in one induction axiom

Automation of Induction

in

Saturation-Based Proof Search

1. Saturation in first-order theorem proving
2. Saturation with induction over term algebras
3. Saturation with induction over integers

Part 3 | Saturation and Induction over integers

- Integers with the standard ordering are not well-founded

Part 3 | Saturation and Induction over integers

- Integers with the standard ordering are not well-founded
- But any set of integers with a lower/upper bound is well-founded

Part 3 | Saturation and Induction over integers

- Integers with the standard ordering are not well-founded
- But any set of integers with a lower/upper bound is well-founded

- We define downward/upward induction schema with symbolic bounds:

$$(F[b_1] \wedge \forall x(b_1 \leq x < b_2 \wedge F[x] \rightarrow F[x+1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow F[x])$$ (upward)

$$(F[b_2] \wedge \forall x(b_1 < x \leq b_2 \wedge F[x] \rightarrow F[x-1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow F[x])$$ (downward)

Part 3 | Saturation and Induction over integers

- Integers with the standard ordering are not well-founded
- But any set of integers with a lower/upper bound is well-founded

- We define downward/upward induction schema with symbolic bounds:

$$(F[b_1] \land \forall x(b_1 \leq x < b_2 \land F[x] \rightarrow F[x+1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow F[x]) \quad \text{(upward)}$$

$$(F[b_2] \land \forall x(b_1 < x \leq b_2 \land F[x] \rightarrow F[x-1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow F[x]) \quad \text{(downward)}$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \lor C}{CNF(\,(L[b_1] \land \forall x(b_1 \leq x < b_2 \land L[x] \rightarrow L[x+1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow L[x]))}$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg\, L[a] \vee C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \;\wedge\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \;\rightarrow\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

Part 3 | Saturation and Induction over integers

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \vee C}{CNF(\,(L[b_1] \wedge \forall x(b_1 \leq x < b_2 \wedge L[x] \rightarrow L[x+1])) \rightarrow \forall x(b_1 \leq x \leq b_2 \rightarrow L[x]))}$$

$$\neg L[b_1] \vee b_1 \leq \sigma \vee \neg b_1 \leq x \vee \neg x \leq b_2 \vee L[x]$$
$$\neg L[b_1] \vee \sigma < b_2 \vee \neg b_1 \leq x \vee \neg x \leq b_2 \vee L[x]$$
$$\neg L[b_1] \vee L[\sigma] \vee \neg b_1 \leq x \vee \neg x \leq b_2 \vee L[x]$$
$$\neg L[b_1] \vee \neg L[\sigma+1] \vee \neg b_1 \leq x \vee \neg x \leq b_2 \vee L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \lor C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \ \land\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \land \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \ \rightarrow \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg L[\mathrm{b}_1] \lor \mathrm{b}_1 \leq \sigma \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \sigma < \mathrm{b}_2 \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \mathrm{L}[\sigma] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \neg \mathrm{L}[\sigma+1] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \lor C}{CNF(\,(L[b_1] \land \forall x(b_1 \le x < b_2 \land L[x] \rightarrow L[x+1])) \rightarrow \forall x(b_1 \le x \le b_2 \rightarrow L[x]))}$$

$$\neg L[b_1] \lor b_1 \le \sigma \lor \neg b_1 \le x \lor \neg x \le b_2 \lor L[x]$$

$$\neg L[b_1] \lor \sigma < b_2 \lor \neg b_1 \le x \lor \neg x \le b_2 \lor L[x]$$

$$\neg L[b_1] \lor L[\sigma] \lor \neg b_1 \le x \lor \neg x \le b_2 \lor L[x]$$

$$\neg L[b_1] \lor \neg L[\sigma+1] \lor \neg b_1 \le x \lor \neg x \le b_2 \lor L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg\, L[a] \lor C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \;\land\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \land \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1]))\;\rightarrow \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg\, L[\mathrm{b}_1] \lor \mathrm{b}_1 \leq \sigma \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg\, L[\mathrm{b}_1] \lor \sigma < \mathrm{b}_2 \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg\, L[\mathrm{b}_1] \lor \mathrm{L}[\sigma] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg\, L[\mathrm{b}_1] \lor \neg \mathrm{L}[\sigma+1] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \vee C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \;\wedge\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \;\rightarrow\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{b}_1 \leq \sigma \vee \neg \mathrm{b}_1 \leq \mathrm{x} \vee \neg \mathrm{x} \leq \mathrm{b}_2 \vee L[x]$$

$$\neg L[\mathrm{b}_1] \vee \sigma < \mathrm{b}_2 \vee \neg \mathrm{b}_1 \leq \mathrm{x} \vee \neg \mathrm{x} \leq \mathrm{b}_2 \vee L[x]$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{L}[\sigma] \vee \neg \mathrm{b}_1 \leq \mathrm{x} \vee \neg \mathrm{x} \leq \mathrm{b}_2 \vee L[x]$$

$$\neg L[\mathrm{b}_1] \vee \neg \mathrm{L}[\sigma+1] \vee \neg \mathrm{b}_1 \leq \mathrm{x} \vee \neg \mathrm{x} \leq \mathrm{b}_2 \vee L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \lor C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \;\land\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \land \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \;\rightarrow\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg L[\mathrm{b}_1] \lor \mathrm{b}_1 \leq \sigma \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \sigma < \mathrm{b}_2 \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \mathrm{L}[\sigma] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

$$\neg L[\mathrm{b}_1] \lor \neg \mathrm{L}[\sigma+1] \lor \neg \mathrm{b}_1 \leq \mathrm{x} \lor \neg \mathrm{x} \leq \mathrm{b}_2 \lor L[x]$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \vee C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \wedge \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \rightarrow \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{b}_1 \leq \sigma \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \sigma < \mathrm{b}_2 \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{L}[\sigma] \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \neg \mathrm{L}[\sigma+1] \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg\, L[a] \vee C}{CNF(\,(\mathrm{L}[\mathrm{b}_1] \;\wedge\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1])) \;\rightarrow\; \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg\, L[\mathrm{b}_1] \vee \quad \mathrm{b}_1 \leq \sigma \quad \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg\, L[\mathrm{b}_1] \vee \quad \sigma < \mathrm{b}_2 \quad \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg\, L[\mathrm{b}_1] \vee \quad \mathrm{L}[\sigma] \quad \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg\, L[\mathrm{b}_1] \vee \neg \mathrm{L}[\sigma+1] \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

Part 3 | Saturation and Induction over integers

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \vee C}{CNF(\ (\mathrm{L}[\mathrm{b}_1]\ \wedge\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1]))\ \rightarrow\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{b}_1 \leq \sigma \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \sigma < \mathrm{b}_2 \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \mathrm{L}[\sigma] \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\neg L[\mathrm{b}_1] \vee \neg \mathrm{L}[\sigma+1] \vee \neg \mathrm{b}_1 \leq \mathrm{a} \vee \neg \mathrm{a} \leq \mathrm{b}_2 \vee C$$

$$\frac{\neg L[a] \vee C \qquad \mathrm{b}_1 \leq \mathrm{a} \qquad \mathrm{a} \leq \mathrm{b}_2}{CNF(\ (\mathrm{L}[\mathrm{b}_1]\ \wedge\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1]))\ \rightarrow\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \lor C}{CNF(\,(L[b_1] \;\land\; \forall x(b_1 \leq x < b_2 \land L[x] \rightarrow L[x+1]))\;\rightarrow\; \forall x(b_1 \leq x \leq b_2 \rightarrow L[x]))}$$

$$\neg L[b_1] \lor \quad b_1 \leq \sigma \quad \lor C$$

$$\neg L[b_1] \lor \quad \sigma < b_2 \quad \lor C$$

$$\neg L[b_1] \lor \quad L[\sigma] \quad \lor C$$

$$\neg L[b_1] \lor \quad \neg L[\sigma+1] \quad \lor C$$

$$\frac{\neg L[a] \lor C \qquad b_1 \leq a \qquad\qquad a \leq b_2}{CNF(\,(L[b_1] \;\land\; \forall x(b_1 \leq x < b_2 \land L[x] \rightarrow L[x+1]))\;\rightarrow\; \forall x(b_1 \leq x \leq b_2 \rightarrow L[x]))}$$

- We introduce integer induction rules, e.g.:

$$\frac{\neg L[a] \vee C}{CNF(\,(\mathrm{L}[\mathrm{b}_1]\ \wedge\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1]))\ \rightarrow \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

$$\frac{\neg L[a] \vee C \qquad \mathrm{b}_1 \leq \mathrm{a} \qquad \mathrm{a} \leq \mathrm{b}_2}{CNF(\,(\mathrm{L}[\mathrm{b}_1]\ \wedge\ \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} < \mathrm{b}_2 \wedge \mathrm{L}[\mathrm{x}] \rightarrow \mathrm{L}[\mathrm{x}+1]))\ \rightarrow \forall \mathrm{x}(\mathrm{b}_1 \leq \mathrm{x} \leq \mathrm{b}_2 \rightarrow \mathrm{L}[\mathrm{x}]))}$$

Part 3 | Saturation and Induction over integers

Example: Prove the assertion over integers p, j and array A

$$(\forall j\ (p \le j < i \rightarrow A[j+1] = A[j]) \quad \wedge \quad \neg\ (i+1 < A.size))$$

$$\rightarrow\ \forall j\ (p \le j < A.size \rightarrow A[j] = A[p])$$

Example: Prove the assertion over integers p, j and array A

$$(\forall j\ (p \leq j < i \rightarrow A[j+1]=A[j]) \quad \wedge \quad \neg\ (i+1 < A.size))$$

$$\rightarrow \forall j\ (p \leq j < A.size \rightarrow A[j]=A[p])$$

Induction scheme (upward):

$$(F[b_1] \ \wedge\ \forall x(b_1 \leq x < b_2 \wedge F[x] \rightarrow F[x+1])\ ) \ \rightarrow\ \forall x(b_1 \leq x \leq b_2 \rightarrow F[x])$$

Part 3 | Saturation and Induction over integers

Example: Prove the assertion over integers p, j and array A

$$(\forall j \, (p \leq j < i \rightarrow A[j+1]=A[j]) \quad \wedge \quad \neg\, (i+1 < A.size))$$

$$\rightarrow \forall j \, (p \leq j < A.size \rightarrow A[j]=A[p])$$

Induction scheme (upward):

$$(F[b_1] \ \wedge \ \forall x(b_1 \leq x < b_2 \wedge F[x] \rightarrow F[x+1]) \,) \ \rightarrow \ \forall x(b_1 \leq x \leq b_2 \rightarrow F[x])$$

Instantiate the scheme:

$F[x]$: $A[x] = A[p]$ $\quad$ b_1: p $\quad$ b_2: $A.size-1$

 Saturation and Induction over integers

Example: Prove the assertion over integers p, j and array A

$$(\forall j\ (p \leq j < i \rightarrow A[j+1]=A[j]) \quad \wedge \quad \neg\,(i+1 < A.size))$$

$$\rightarrow\ \forall j\ (p \leq j < A.size \rightarrow A[j]=A[p])$$

Induction scheme (upward):

$$(F[b_1]\ \wedge\ \forall x(b_1 \leq x < b_2 \wedge F[x] \rightarrow F[x+1])\ ) \ \rightarrow\ \forall x(b_1 \leq x \leq b_2 \rightarrow F[x])$$

Instantiate the scheme:

$F[x]$: $A[x] = A[p]$ $\qquad$ b_1: p $\qquad$ b_2: $A.size-1$

Saturation and integer induction rules + resolution ✓

Problem set	Total count	CVC4	Z3	Vampire*	new compared to Vampire	new compared to Vampire, CVC4 & Z3
LIA	607	553	435	214	10	1
UFLIA	10137	7002	6705	5796	99	44

Part 3 | Saturation and Induction over integers

	Count	ACL2	CVC4	Vampire*
array	84	5	26	75
sum	12	0	5	11
power	24	0	0	20
total	120	5	31	106
uniquely solved	-	0	3	75

	Count	ACL2	CVC4	Vampire*
array	84	5	26	75
sum	12	0	5	11
power	24	0	0	20
total	120	5	31	106
uniquely solved	-	0	3	75

Array properties

Assumption: $\forall j \in \mathbb{Z}.(b_1 \leq j < b_2 \rightarrow \mathrm{v}(j+1) = \mathrm{v}(j))$
Goal: $\forall j \in \mathbb{Z}.(b_1 \leq j \leq b_2 \rightarrow \mathrm{v}(j) = \mathrm{v}(b_1))$

Saturation and Induction over integers

	Count	ACL2	CVC4	Vampire*
array	84	5	26	75
sum	12	0	5	11
power	24	0	0	20
total	120	5	31	106
uniquely solved	-	0	3	75

Sum properties

$$\forall x \forall y\ (x \leq y \rightarrow 2 * sum(x,y) = y*(y+1) - x*(x-1))$$

	Count	ACL2	CVC4	Vampire*
array	84	5	26	75
sum	12	0	5	11
power	24	0	0	20
total	120	5	31	106
uniquely solved	-	0	3	75

Power properties

$\forall x \forall y \forall e (1 \leq e \rightarrow power(x*y,e)=power(x,e)*power(y,e))$

Conclusions | Saturation and Induction

Induction in saturation-based proof search

- Induction rules in saturation
- Proving inductive properties from software analysis, math, …

Conclusions | Saturation and Induction

<u>dataype</u> nat = zero | succ <u>of</u> nat

add zero x = x

add (succ x) y = succ (add x y)

<u>assert</u> *(∀ x) (∀ y) (add x y) = (add y x)*

<u>*assume*</u> $0 \leq p < A.size$

```
i:=p;
while (i+1< A.size) do
    A[i+1]:=A[i];
    i:=i+1;
    invariant (∀j∈ℤ)(p≤j<i ⇒ A[j+1]=A[j])
end do
assert (∀j∈ℤ)(p≤j<A.size ⇒ A[j]=A[p])
```

Proving inductive properties

Conclusions | Saturation and Induction

Induction in saturation-based proof search

- Induction rules in saturation
- Proving inductive properties from software analysis, math, …

Further work on

- Guiding search (when to apply induction, reduce applications of induction, …)
- Use rewriting (extensions on recursive function definitions, …)
- Lemma generation (consequence finding, generalizations, …)

FWF
Der Wissenschaftsfonds.
erc
European Research Council
Supporting top researchers from anywhere in the world
W|W|T|F
VIENNA SCIENCE AND TECHNOLOGY FUND
amazon webservices™