HORN ABDUCTION

PRESENTED BY RODY KERSTEN

N BJØRNER, D JOVANOVÍC, T LEPOINT, P RÜMMER, M SCHÄF

Horn Clauses in Verification

```
public class MyExample {
  public static void main(String[] args) {
    final int len = 10;
    int x=0, y=0;
    while (x<len) {
      x+=1; y+=2;
    }
    assert x+y==3*len;
  }
}
```

$$\text{entry}(x,y,\text{len}) \leftarrow \text{true}$$

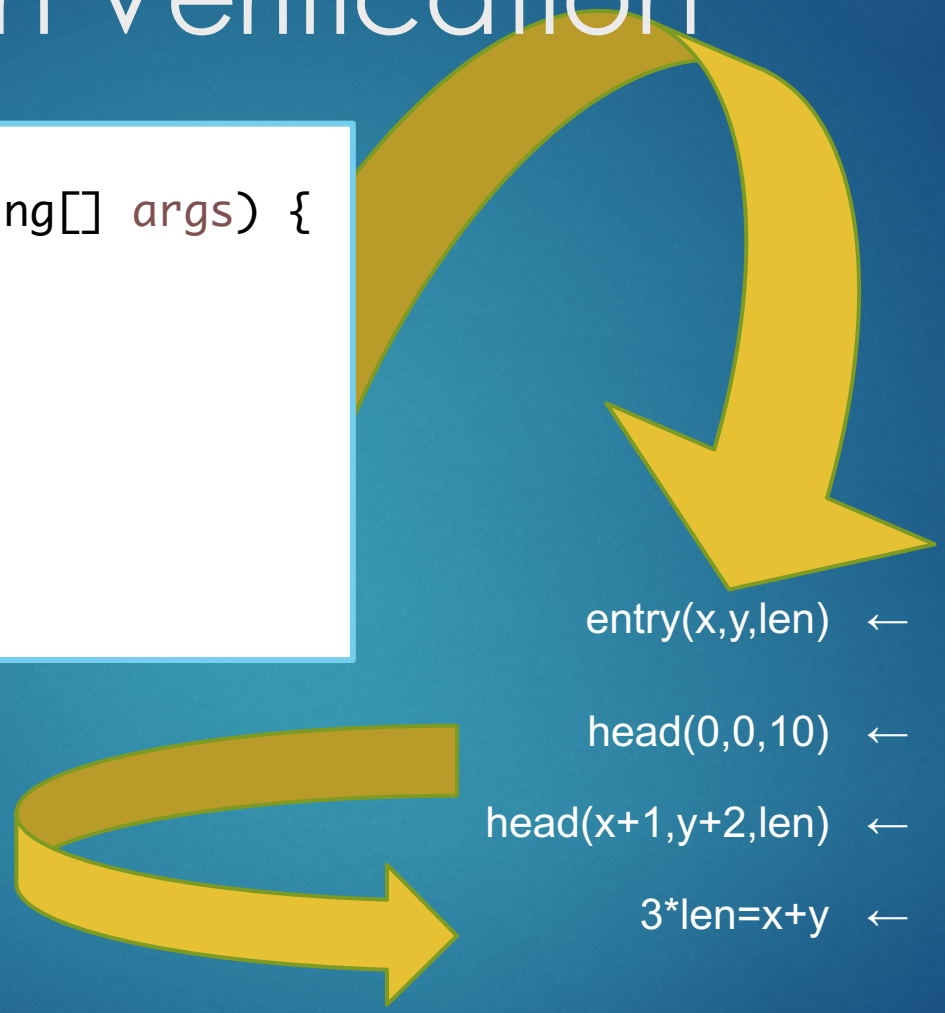
$$\text{head}(0,0,10) \leftarrow \text{entry}(x,y,\text{len})$$

$$\text{head}(x+1,y+2,\text{len}) \leftarrow x<\text{len} \wedge \text{head}(x,y,\text{len})$$

$$3*\text{len}=x+y \leftarrow x=\text{len} \wedge \text{head}(x,y,\text{len})$$

Horn Clauses in Verification

```
public class MyExample {
  public static void main(String[] args) {
    final int len = 10;
    int x=0, y=0;
    while (x<len) {
      x+=1; y+=2;
    }
    assert x+y==3*len;
  }
}
```



$$\text{entry}(x,y,len) \leftarrow \text{true}$$

$$\text{head}(0,0,10) \leftarrow \text{entry}(x,y,len)$$

$$\text{head}(x+1,y+2,len) \leftarrow x<len \wedge \text{head}(x,y,len)$$

$$3*len=x+y \leftarrow x=len \wedge \text{head}(x,y,len)$$

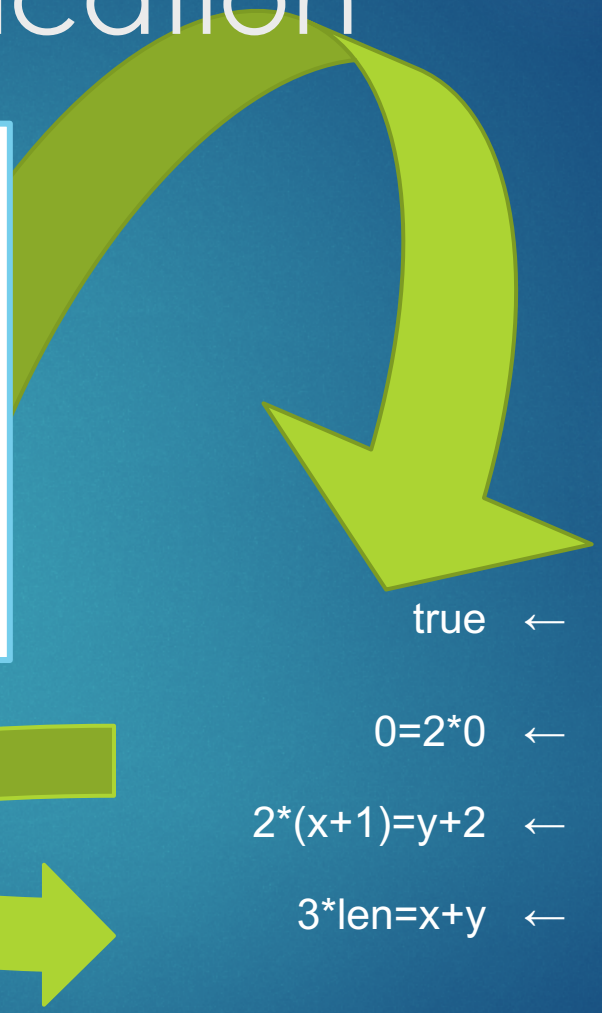
Horn solver searches assignments:

entry(x,y,len): true

head(x,y,len): y=2*x

Horn Clauses in Verification

```
public class MyExample {
  public static void main(String[] args) {
    final int len = 10;
    int x=0, y=0;
    while (x<len) {
      x+=1; y+=2;
    }
    assert x+y==3*len;
  }
}
```



true ← true

0=2*0 ← true

2*(x+1)=y+2 ← x<len ∧ (y=2*x)

3*len=x+y ← x=len ∧ (y=2*x)

Horn solver searches assignments:

entry(x,y,len): true

head(x,y,len): y=2*x

What do we do if the solver fails?

33 lines (32 sloc) | 5.6 KB

Raw | Blame | History

```
(set-info :origin "NTS benchmark converted to SMT-LIB2 using Eldarica (http://lara.epfl.ch/w/eldarica)")
(set-logic HORN)
(declare-fun main_q2 (Int Int Int Int Int Int) Bool)
(declare-fun main_qf () Bool)
(declare-fun main_q0 (Int Int Int Int Int Int) Bool)
(declare-fun main_q1 (Int Int Int Int Int Int) Bool)
(declare-fun search_q6 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q4 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q5 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q3 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q2 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q7 (Int Int Int Int Int Int) Bool)
(declare-fun search_q1 (Int Int Int Int Int Int Int Int Int Int) Bool)
(declare-fun search_q0 (Int Int Int Int Int Int Int Int Int Int) Bool)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int))(=>(and (main_q2 ?A ?B ?C ?D ?E ?F)(= ?D 1)) main_qf)))
(assert(not (exists((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int))(and (main_q2 ?A ?B ?C ?D ?E ?F)(not (= ?D 1))))))
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int))(=>(and (main_q0 ?A ?B ?C ?G ?H ?I)(and
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int))(=>(and (and (= ?A ?D) (= ?B ?E)) (= ?C ?F)) (main_q0 ?A ?B ?C
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(assert(forall((?A Int)(?B Int)(?C Int)(?D Int)(?E Int)(?F Int)(?G Int)(?H Int)(?I Int)(?J Int)(?K Int)(?L Int)(?M Int)(?N Int)
(check-sat)
```

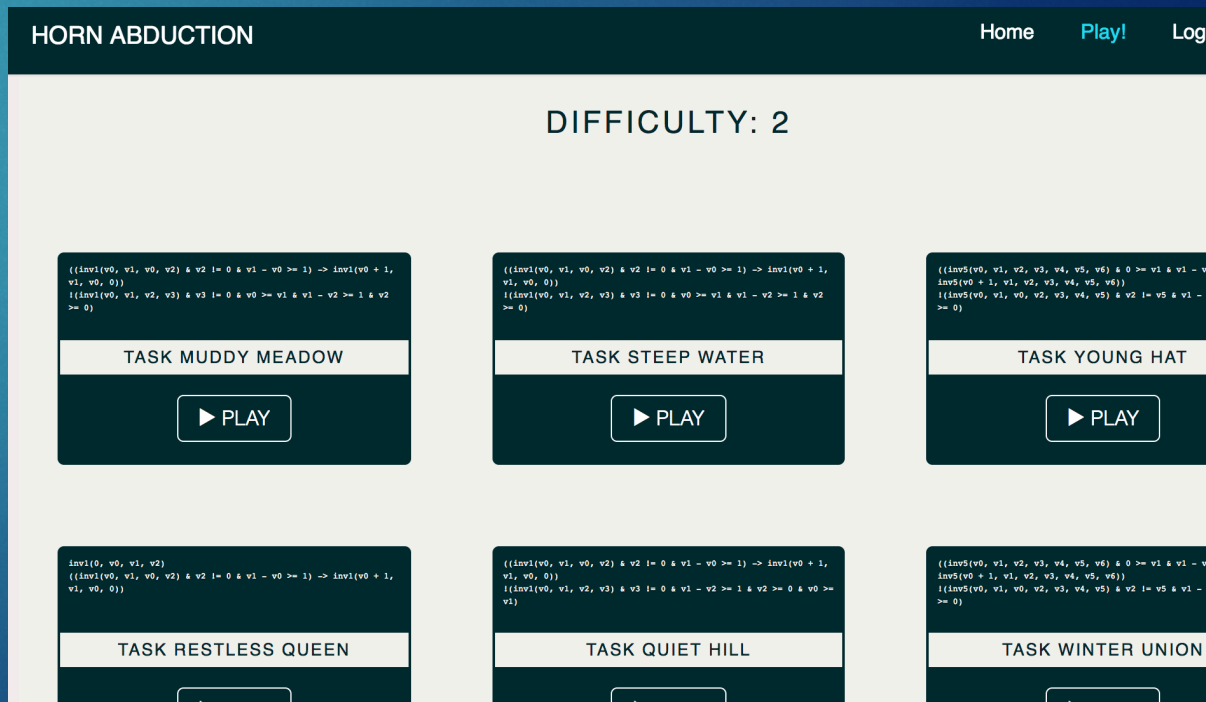
Crowdsourcing 101

- Problem has to be easily dividable into crowdsourceable chunks.
- Problem has to be hard for machines and doable for humans.
- Solution has to be easily checkable by a machine.
- Difficulty has to be adjustable to motivate users.

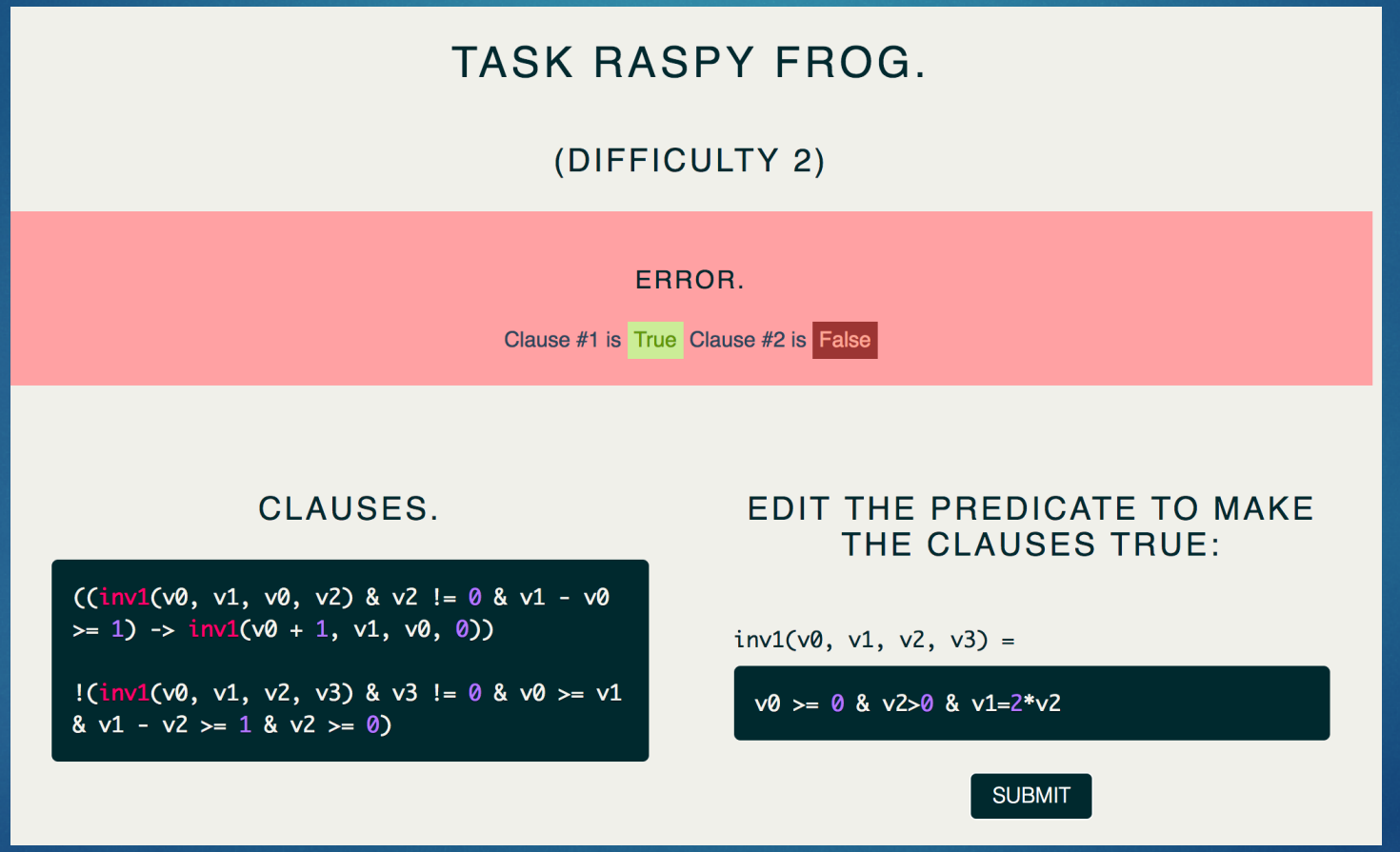
Crowdsourcing Horn solving http://www.horn-abduction.org

Setup

- Run Eldarica on SVComp Horn problems
- Stop Eldarica after 30s and record the current predicate assignments
- Turn a subset of the Horn clauses and predicate assignments into a crowdsourcing problem:
  - User has to improve the current assignments to make all clauses (in the subset) valid.
  - User must not use trivial clauses (e.g., false)



Crowdsourcing tasks



Concluding remarks

- First experiments show that it is surprisingly easy to get users excited to participate (over 700 solutions in 2 weeks by just posting it on Facebook).
- Some people even wrote bots with their own solvers.
- Getting the subdivision of the Horn clauses right is much harder than anticipated: picking a bad subset prevents users from even expressing a useful invariant.
- Style over content: for Crowdsourcing, presentation is everything. If the interface is not appealing and responsive, there will be no crowd to solve problems.

Thanks

The game: http://www.horn-abduction.com

The code: https://github.com/crowdhorn/horngame

Please feel free to fork and improve!