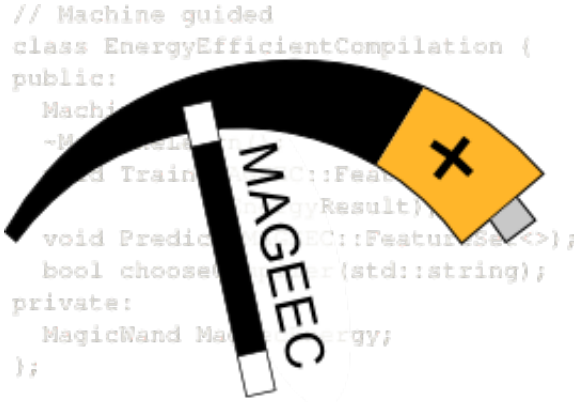
Declarative Machine Learning for Energy Efficient Compiler Optimizations

Craig Blackmore

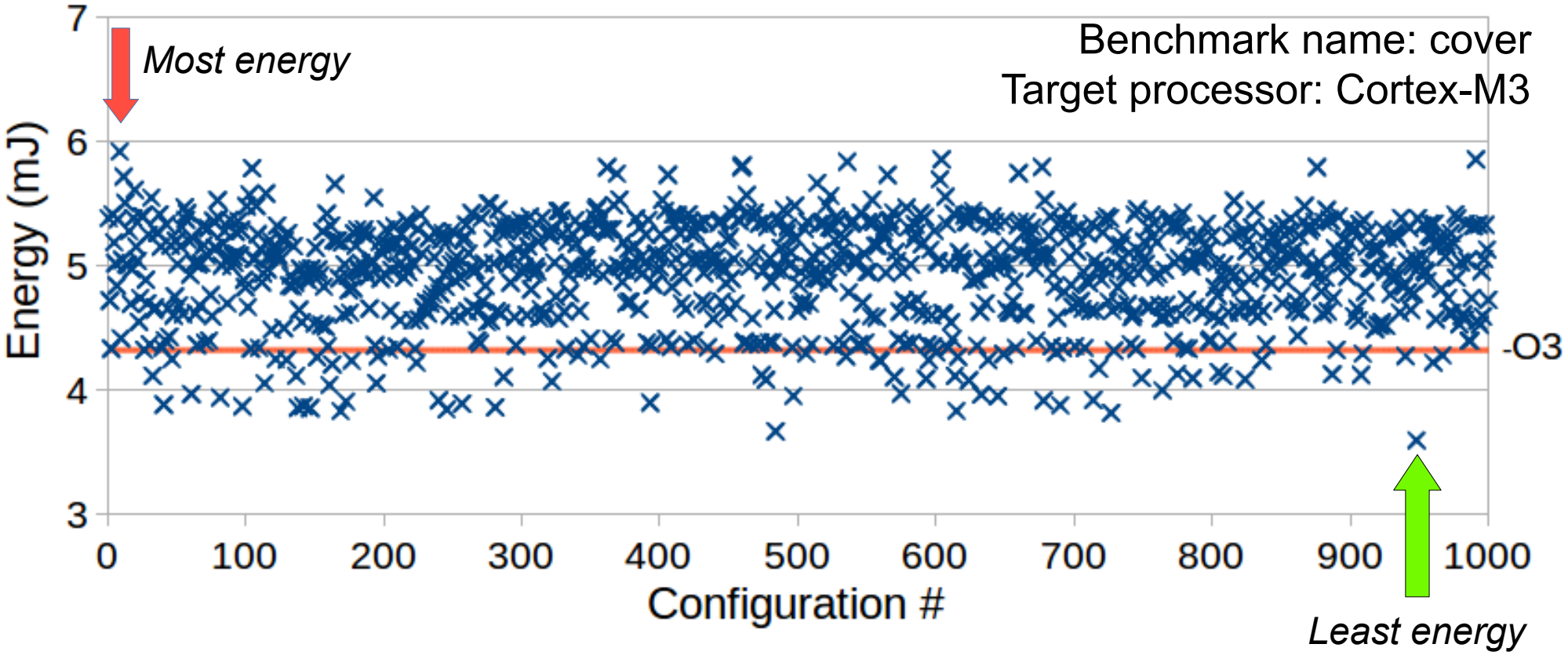
Kerstin Eder

Oliver Ray

University of BRISTOL



1000 Random Configurations



✗ Exhaustive search infeasible

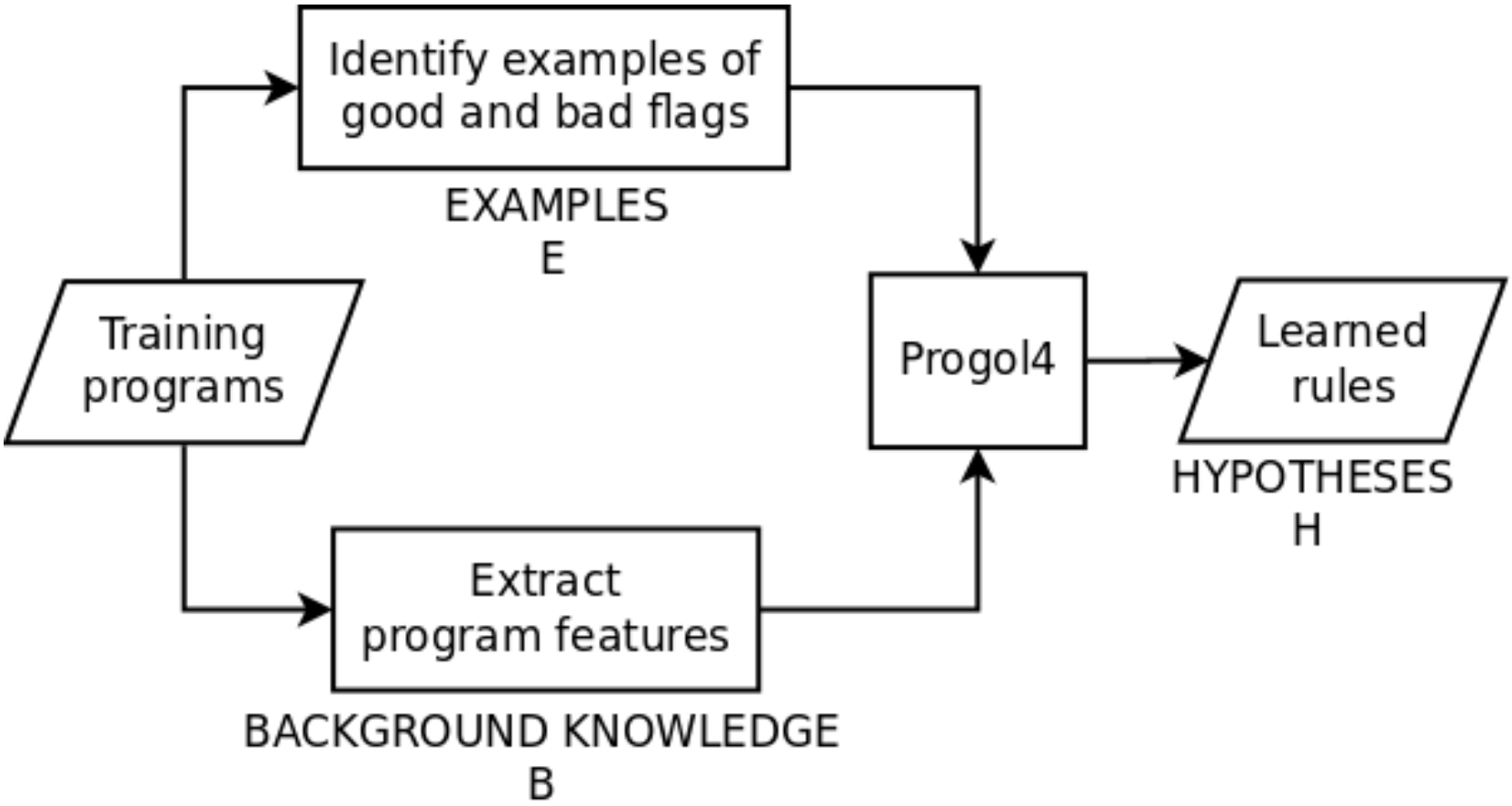
✗ Random sampling (iterative compilation) slow

✔ **Predictive approaches**

Inductive Logic Programming

- **Declarative inductive learning method**
- Requires:
  - Factual examples *E* represented as components together with their relationships, and
  - Background knowledge *B* describing these relations.
- Finds a single (or multiple) hypothesis *H*
  - expressed in terms of the relations given in *B*
  - such that every positive and (ideally)
  - no negative example in *E* is covered by *H*.

ILP for MAGEEC



Specific examples → General Rules

Identifying Good/Bad Flags

1000 random configurations for each benchmark

Good flag = appears frequently in top configurations

Bad flag = not a good flag ☺

Examples *E* for Progol:

goodFlag(program1, -finline-functions).	Positive Example
:- badFlag(program1, -finline-functions).	Negative Example

Feature Extraction

Prolog queries regarding the program features + utility functions:

Ft1 - Number of basic blocks

featlstn.P : ft(ft1,N) :- findall(B,bb_p(B),L), count_lst(L,N).

Ft5 - Number of basic blocks with a single predecessor

featlstn.P :

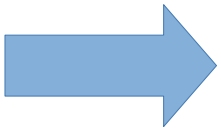
```
edge_dest_pr2(B,N) :- bb_p(B), findall(E,edge_dest(E,B),L),
count_lst(L,N).
edge_dest_pr2_sel1(B) :- edge_dest_pr2(B,N), N = 1.
ft(ft5,N) :- findall(B,edge_dest_pr2_sel1(B),L), count_lst(L,N).
```

Feature Extraction

- Ft15 - Number of basic blocks with number of instructions greater than 500
  - featlstn.P :
    - bb_stmt_in(B,ST) :- bb_stmt_f(B,ST).
    - bb_stmt_in(B,ST) :- bb_stmt_n(B,_,ST).
    - bb_stmt_in_pr1(B,N) :- bb_p(B), findall(ST,bb_stmt_in(B,ST),L), count_lst(L,N).
    - bb_stmt_in_pr1_sel3(B) :- bb_stmt_in_pr1(B,N), N > 500.
    - ft(ft15,N) :- findall(B,bb_stmt_in_pr1_sel3(B),L), count_lst(L,N).

Background Knowledge

Feature 1 = 0.5
Feature 2 = 0.7
Feature 3 = 0.32
...



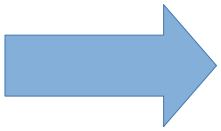
ft(ft1, program1, 0.5).
ft(ft2, program1, 0.7).
ft(ft3, program1, 0.32).
...

Background relations:

quartile(P,Ft,1) :- ft(Ft,P,N), qt1(Ft,Q1), N=<Q1.
quartile(P,Ft,2) :- ft(Ft,P,N), qt1(Ft,Q1), qt2(Ft,Q2), N>Q1, N=<Q2.
quartile(P,Ft,3) :- ft(Ft,P,N), qt2(Ft,Q2), qt3(Ft,Q3), N>Q2, N=<Q3.
quartile(P,Ft,4) :- ft(Ft,P,N), qt3(Ft,Q3), N>Q3.

Background Knowledge

Feature 1 = 0.5
Feature 2 = 0.7
Feature 3 = 0.32
...



ft(ft1, program1, 0.5).
ft(ft2, program1, 0.7).
ft(ft3, program1, 0.32).
...

Background relations:

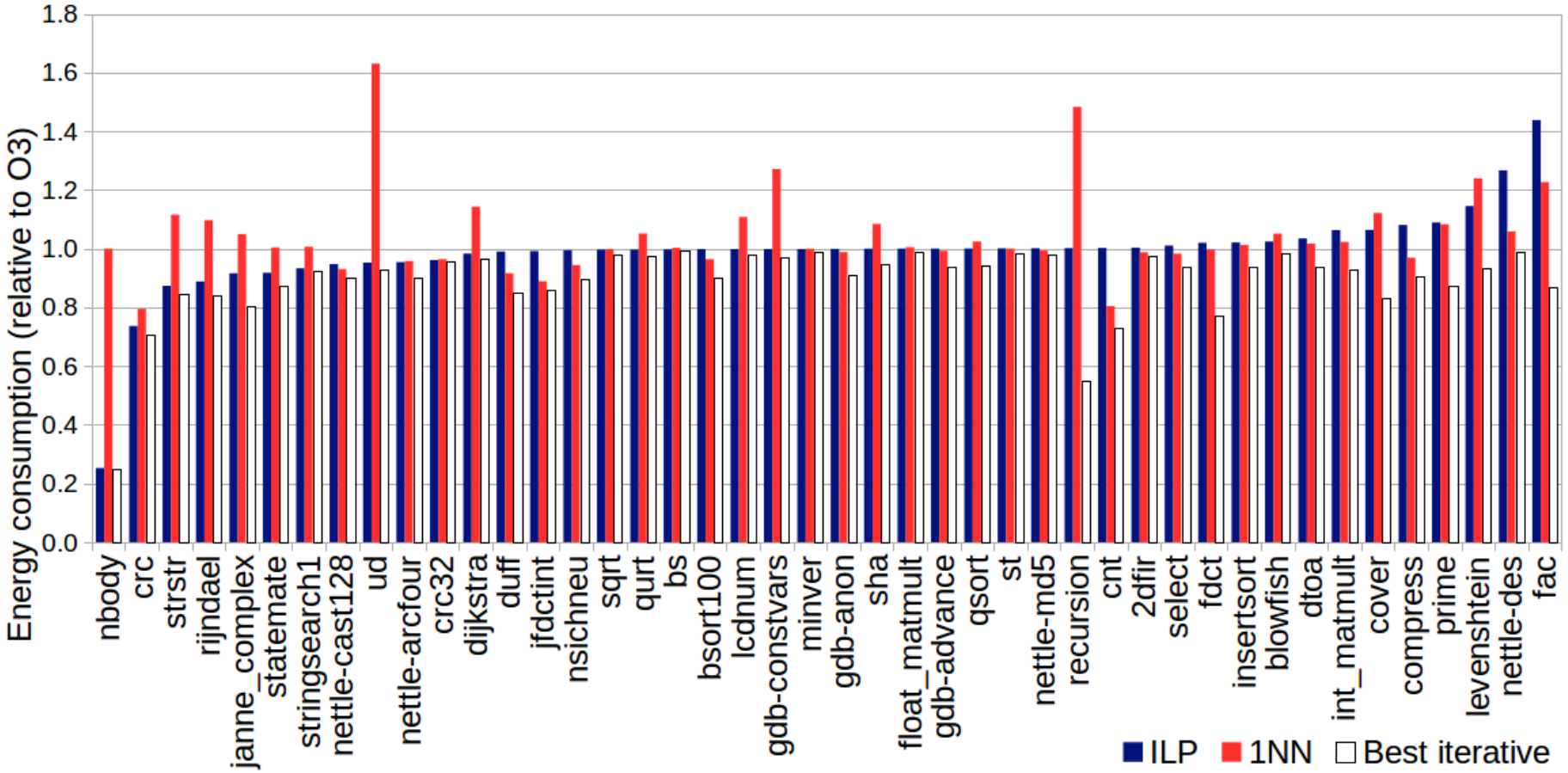
quartile(P,Ft,1) :- ft(Ft,P,N), qt1(Ft,Q1), N=<Q1.
quartile(P,Ft,2) :- ft(Ft,P,N), qt1(Ft,Q1), qt2(Ft,Q2), N>Q1, N=<Q2.
quartile(P,Ft,3) :- ft(Ft,P,N), qt2(Ft,Q2), qt3(Ft,Q3), N>Q2, N=<Q3.
quartile(P,Ft,4) :- ft(Ft,P,N), qt3(Ft,Q3), N>Q3.

Learning Results

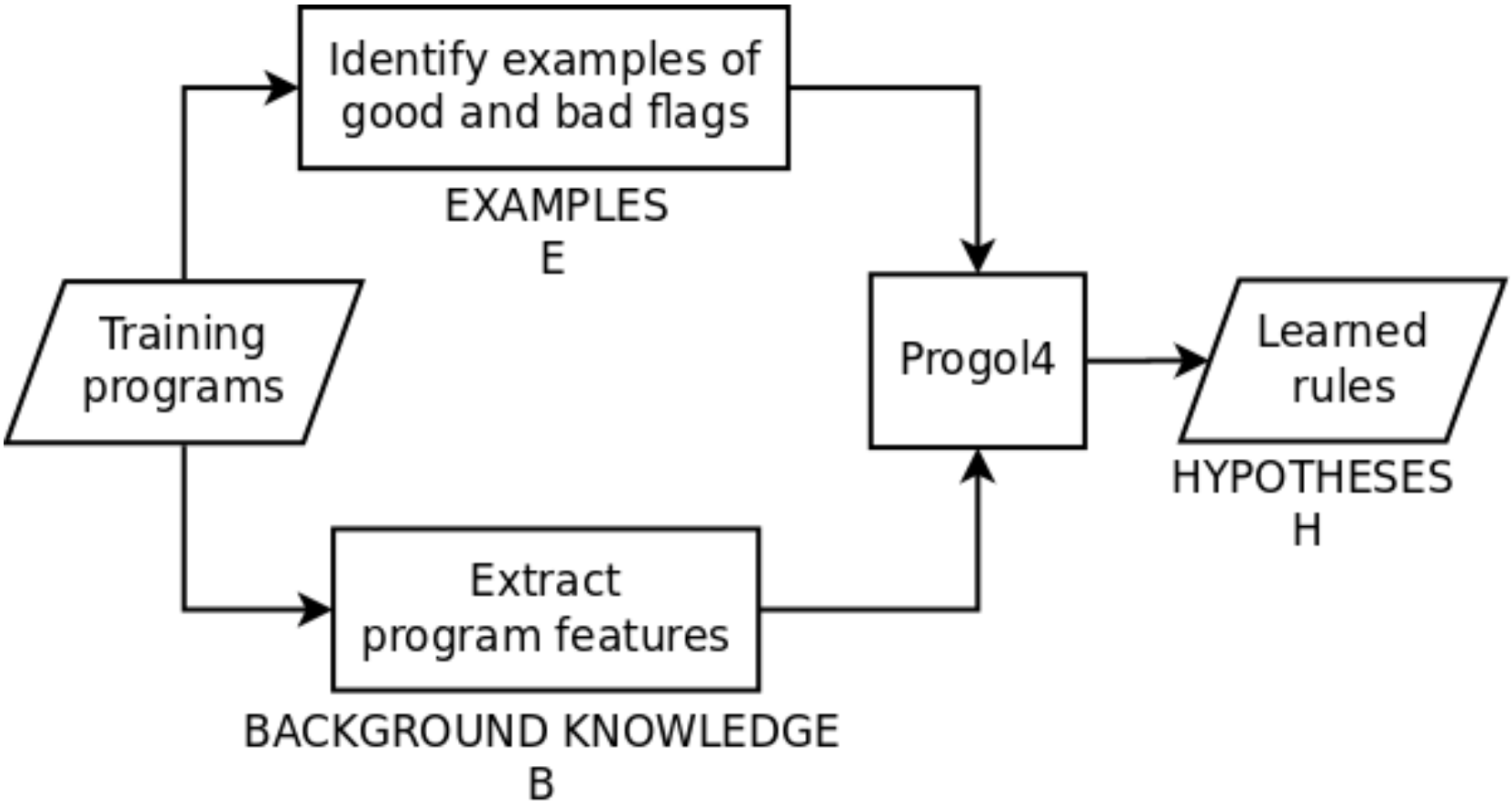
goodFlag(A,-finline-small-functions).

badFlag(A,-fsection-anchors) :- quartile(A,ft22,3).

Preliminary Results: Leave-One-Out Cross Validation

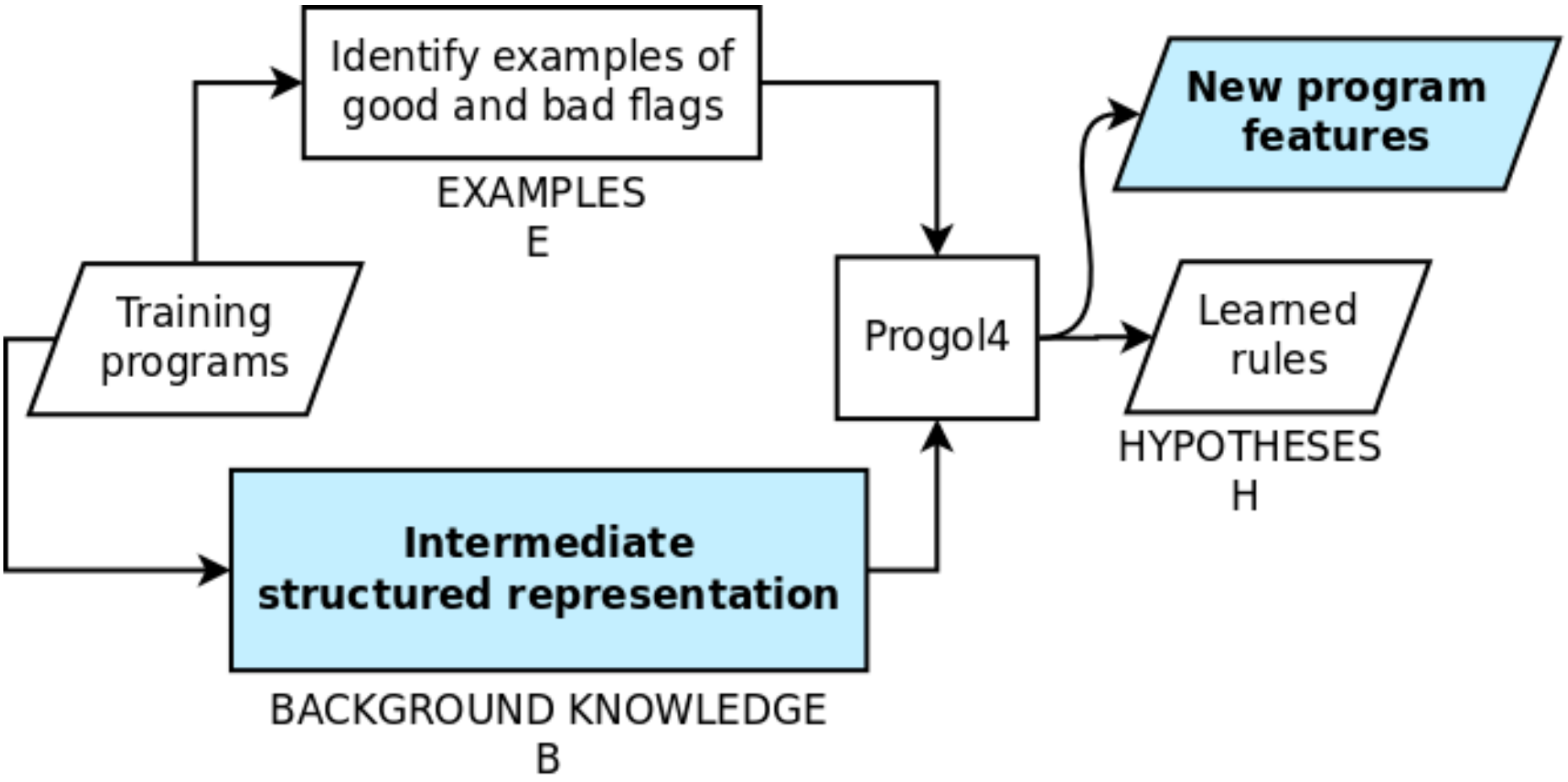


ILP for MAGEEC



Specific examples → General Rules

Future Work



Specific examples → General Rules

Questions?

Craig Blackmore, Kerstin Eder and Oliver Ray
"Declarative Machine Learning for Energy Efficient Compiler Optimization"

Accepted for presentation at the
24th International Conference on Inductive Logic Programming
14-16 September 2014