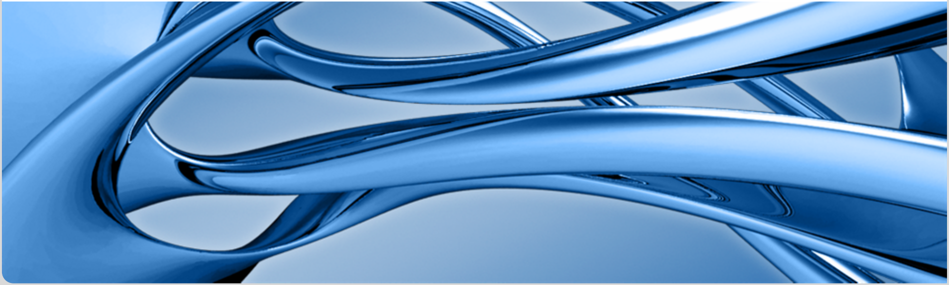
Karlsruhe Institute of Technology

Beans and Eggs - A Glucose Hack

Markus Iser, KIT

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)



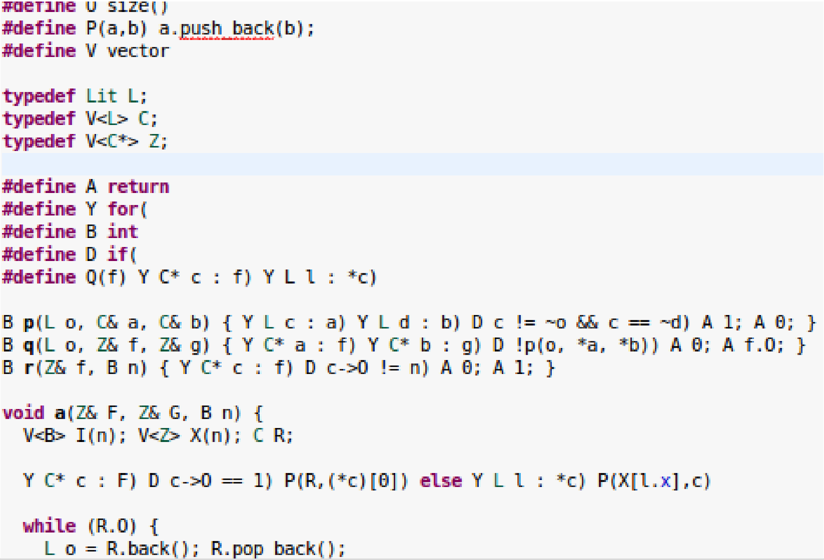
KIT – University of the State of Baden-Wuerttemberg and
National Research Center of the Helmholtz Association

www.kit.edu

Gate Recognition before Solving

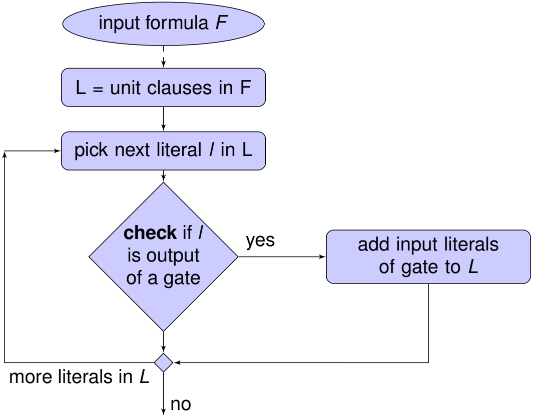
- *Idea:* Run gate extraction algorithm before the solve process and make the solver use the extracted information
- *Challenge:* Perform gate extraction in less than 1000 characters of code (at first healthy, later ugly)
- Leave space for code that properly uses that information: In our case we do a variant of input branching (activity initialization)

The other side of the character limit



```
#define O size()
#define P(a,b) a.push_back(b);
#define V vector

typedef Lit L;
typedef V<L> C;
typedef V<C*> Z;


#define A return
#define Y for(
#define B int
#define D if(
#define Q(f) Y C* c : f) Y L l : *c)

B p(L o, C& a, C& b) { Y L c : a) Y L d : b) D c != ~o && c == ~d) A 1; A 0; }
B q(L o, Z& f, Z& g) { Y C* a : f) Y C* b : g) D !p(o, *a, *b)) A 0; A f.O; }
B r(Z& f, B n) { Y C* c : f) D c->O != n) A 0; A 1; }

void a(Z& F, Z& G, B n) {
  V<B> I(n); V<Z> X(n); C R;

  Y C* c : F) D c->O == 1) P(R,(*c)[0]) else Y L l : *c) P(X[l.x],c)

  while (R.O) {
    L o = R.back(); R.pop back();
```

Core Gate Extraction Algorithm


input formula *F*
L = unit clauses in F
pick next literal *l* in L
check if *l* is output of a gate
yes
add input literals of gate to *L*
more literals in *L*
no

- stripped clause selection loop
- check is blocked set check
- several optimizations regarding datastructures
- self-contained (STL)

Iser, Manthey, Sinz - Recognition of Nested Gates in CNF Formulas (SAT 2015)

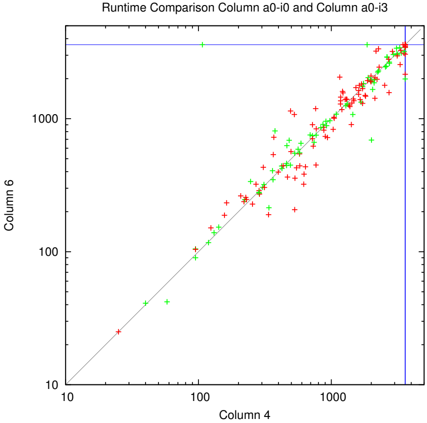
How to use the information

- variant of input branching
- initialize activity values for input variables
- favour frequent input variables (occurrence count)

We already had positive results with the method on satisfiable problems in the relational model finder Kodkod (Alloy) in the past (Iser, Sinz, Taghdiri, 2012). Back then we extracted the information from Kodkod's AST.

Positive Result

In initial experiments with the SAT 2014 Benchmark set we could solve 4 problems more than the reference glucose (175 vs. 179) within 3600*s* timeout.



Negative Result

- Scaling activity initialization values between 1 and 10 gave very mixed results. Intensification of the initialization by scaling up the value for some problems was uncorrelated with feasibility of all the problems
- **However**, there were some types of problems where the positive effect was more reliable: mostly Rintanen's *planning problems*. Also with an extended set of benchmarks including more of these planning problems the positive effect on runtime remained (1 timeout less, several extreme speedups, from close to timeout to few seconds)
- **BTW**, LeBerre's *strippacking /korf problems* can also be solved faster with input initialization (also korf18), but it works without gate recognition. Just a simple occurrence count for intialization is enough

Recent and Future Work

- Use extracted structure for SAT-sweeping (diploma thesis of Felix Kutzner, just submitted)
- Write STL version of glucose (Codname: candy-kingdom, available on GitHub)
- Integrate gate-extraction algorithm and usage parameters in candy-kingdom (August 1st, watch out)