CS 292C Computer-Aided Reasoning for Software

Lecture 6: Applications of SAT

Yu Feng

Fall 2019

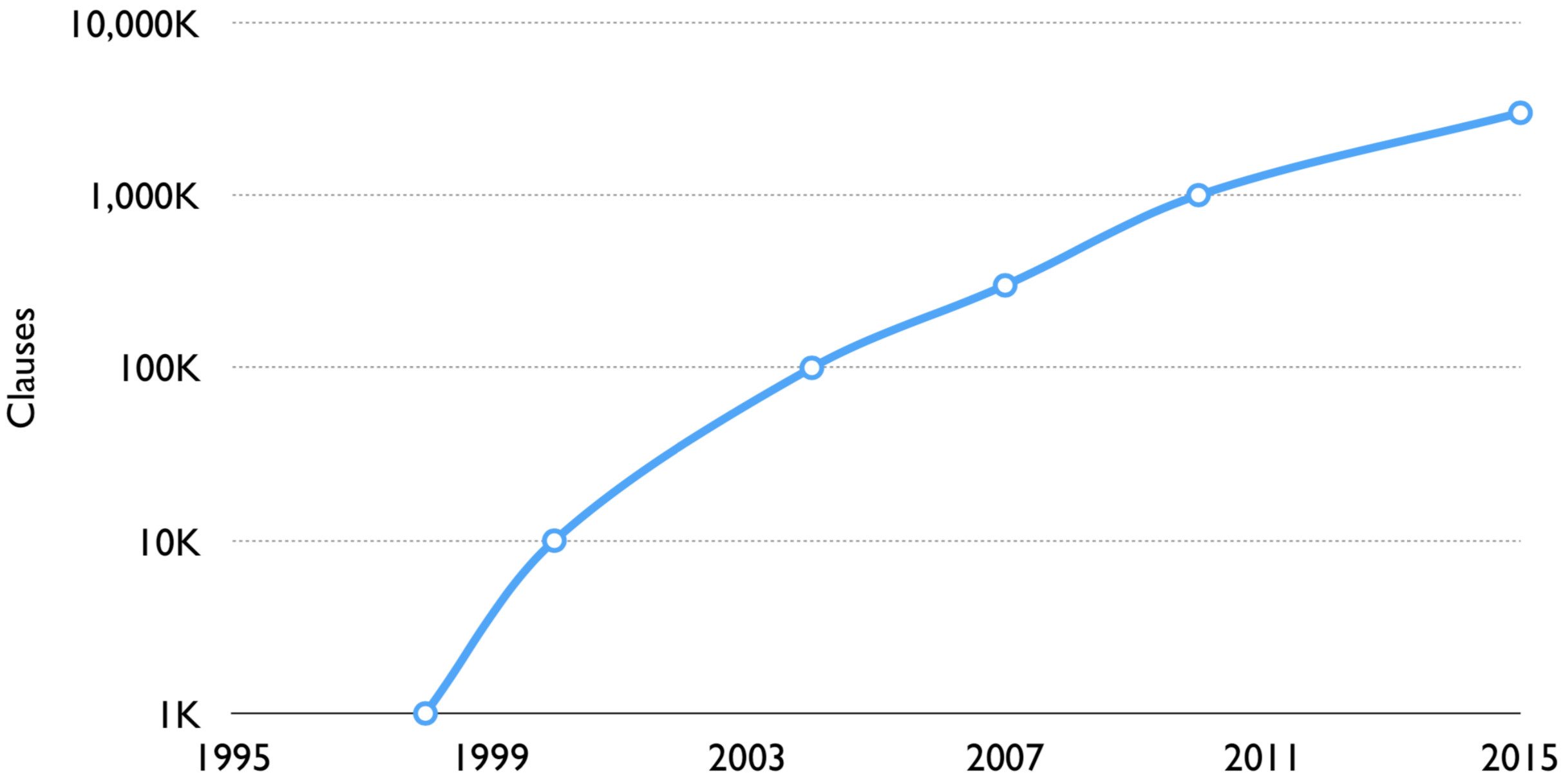
Summary of previous lecture

- 1st homework is due now
- 2nd paper review is also due now
- CDCL algorithm

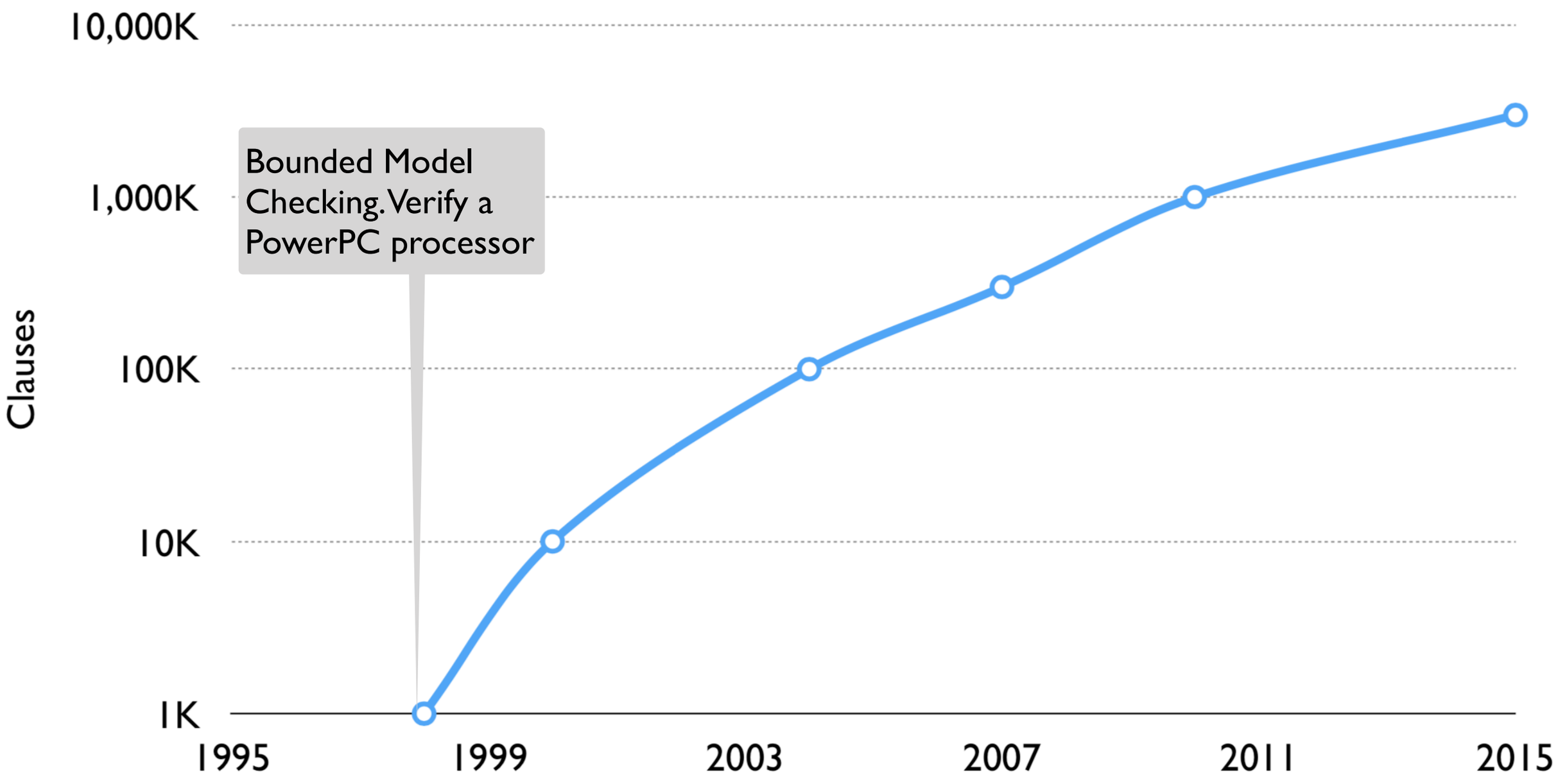
Outline of this lecture

- Practical applications of SAT
- Variants of the SAT problem

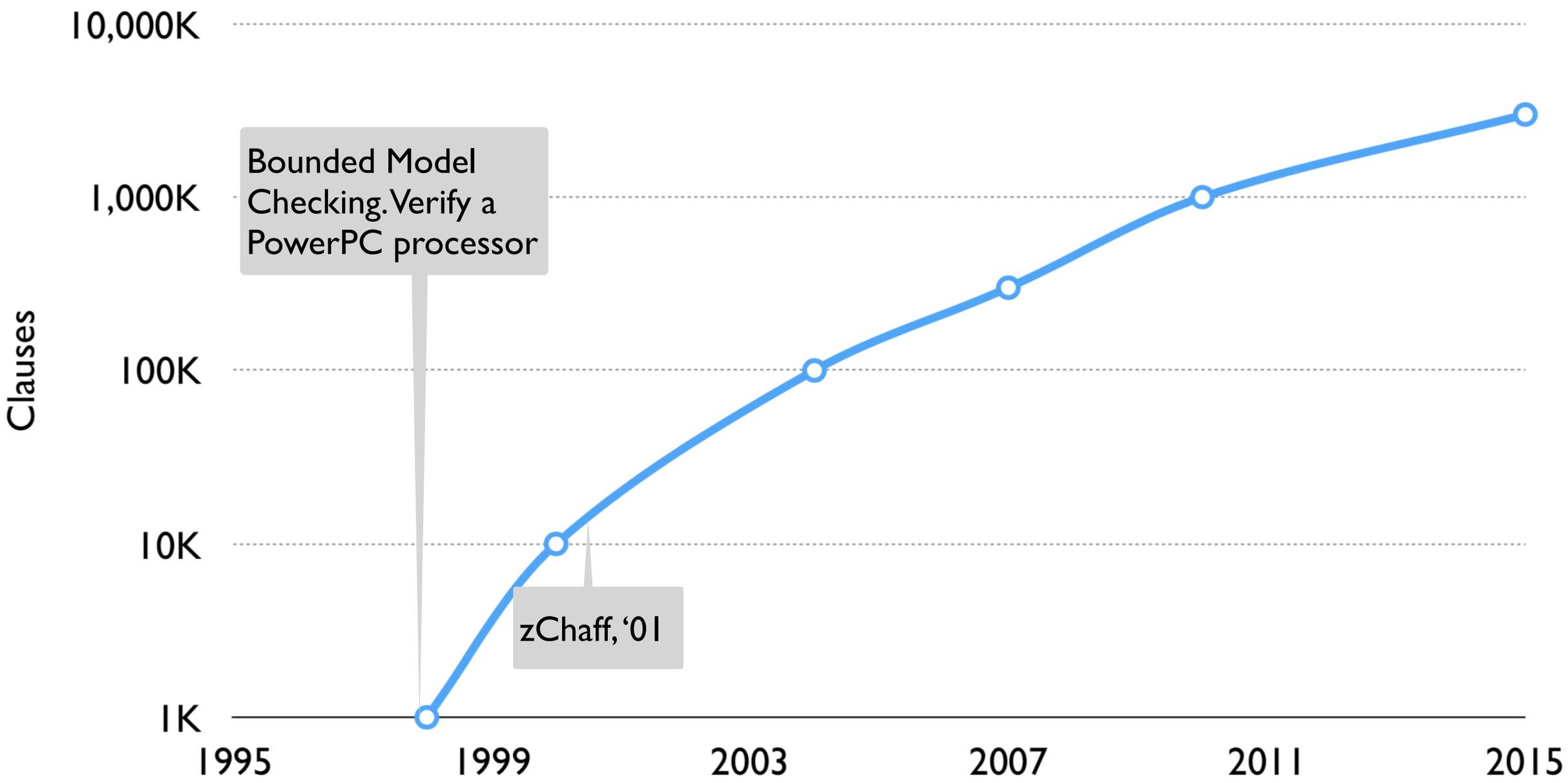
SAT solving and its applications



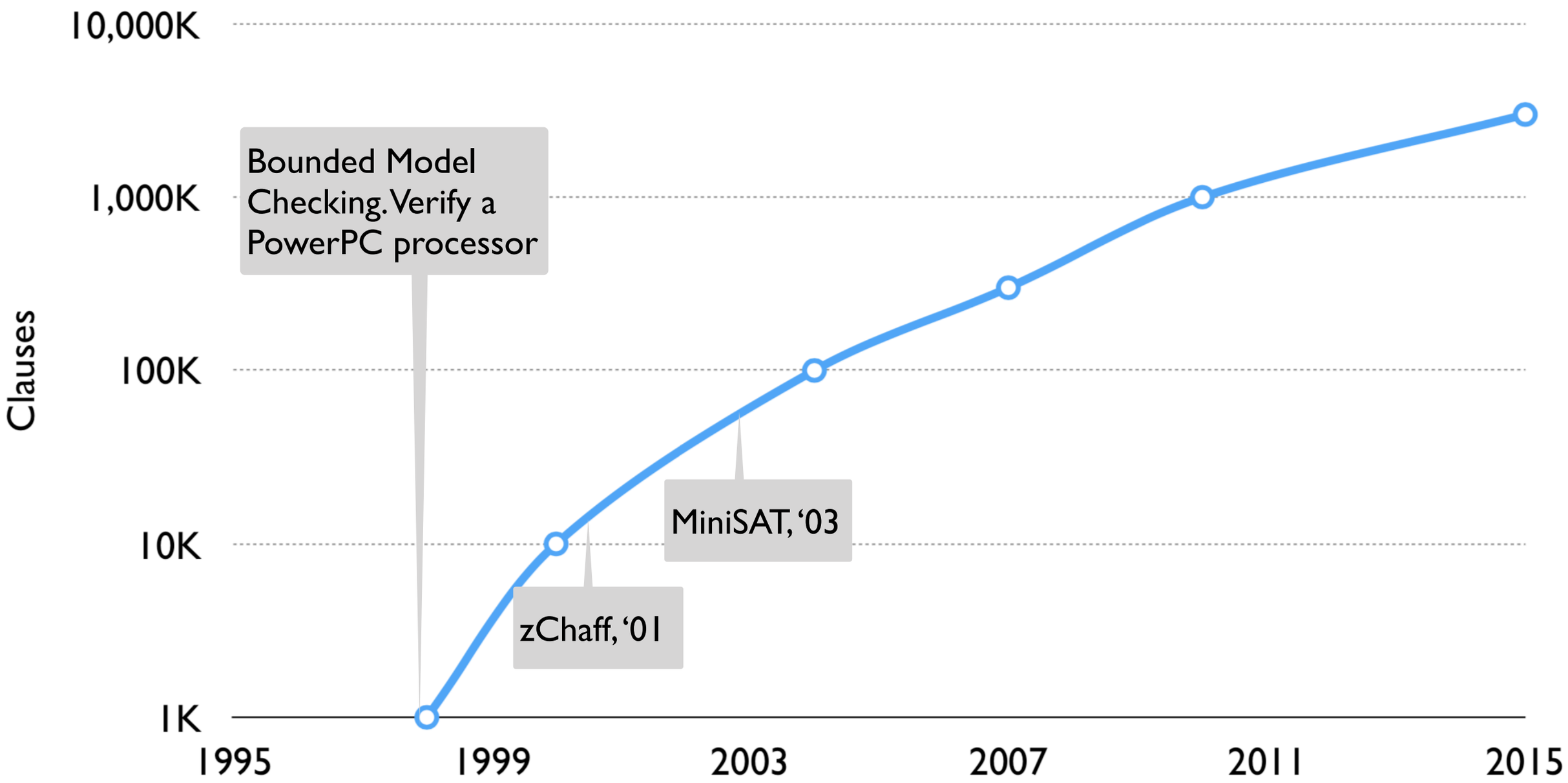
SAT solving and its applications



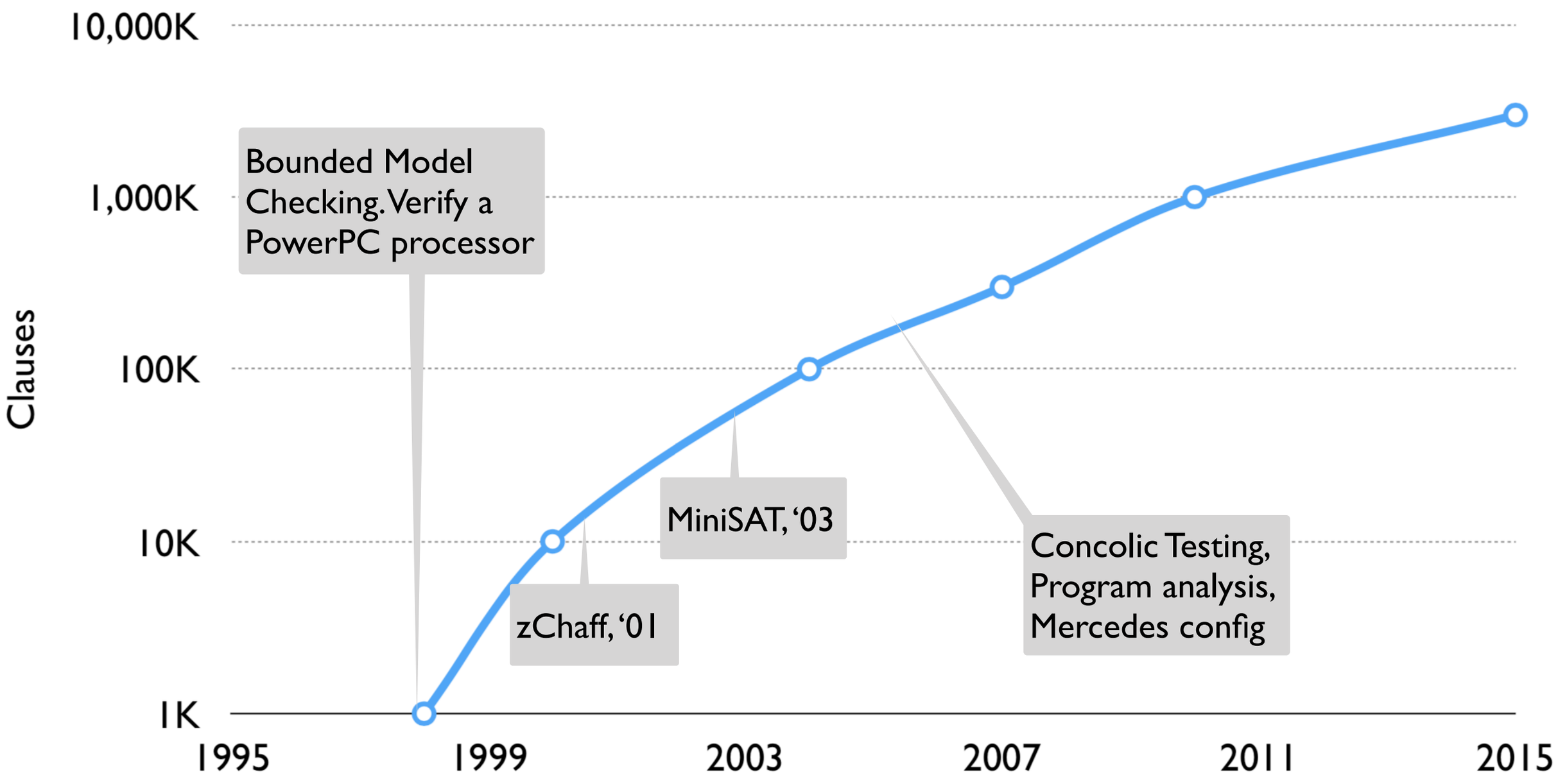
SAT solving and its applications



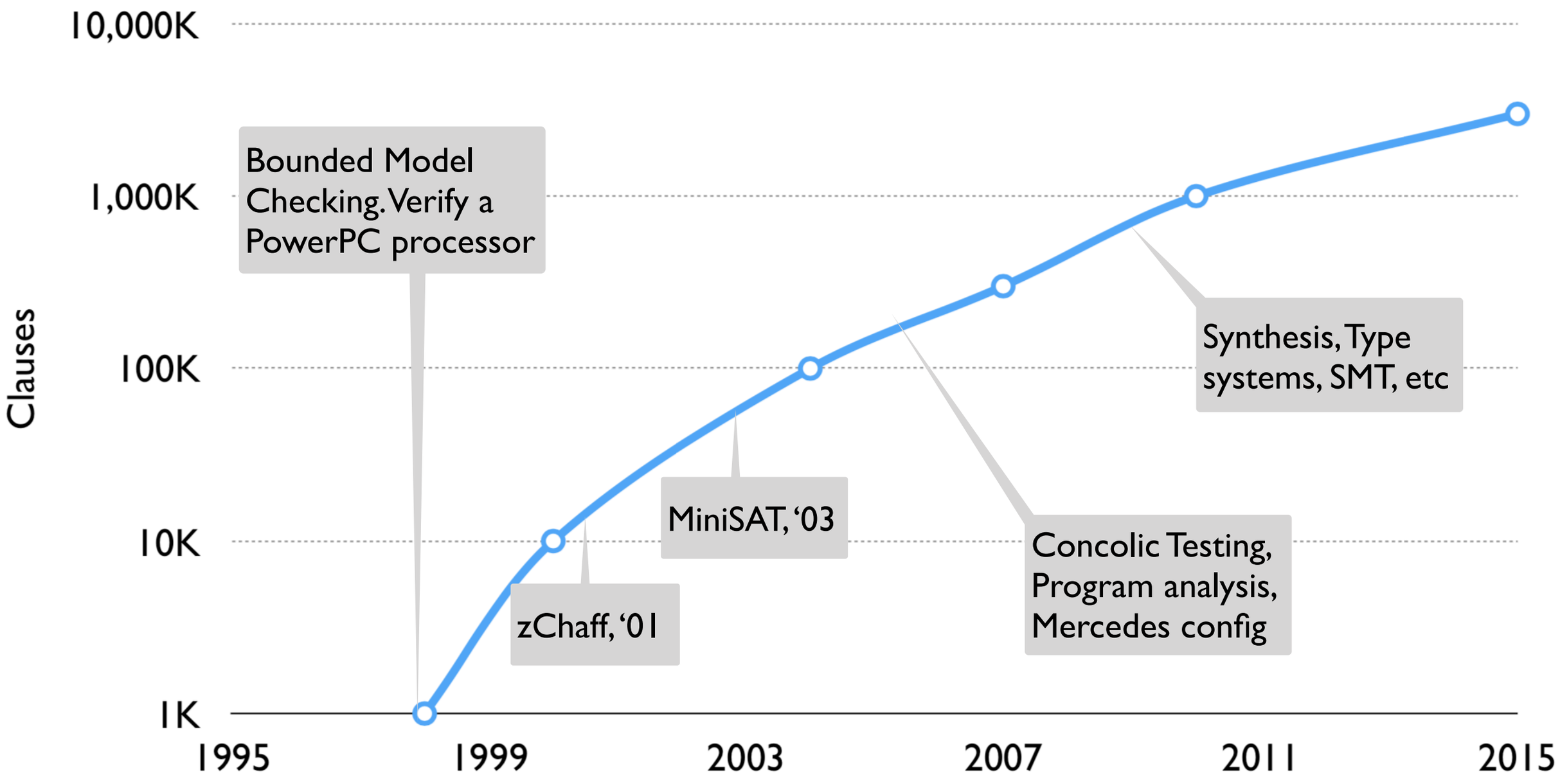
SAT solving and its applications



SAT solving and its applications



SAT solving and its applications

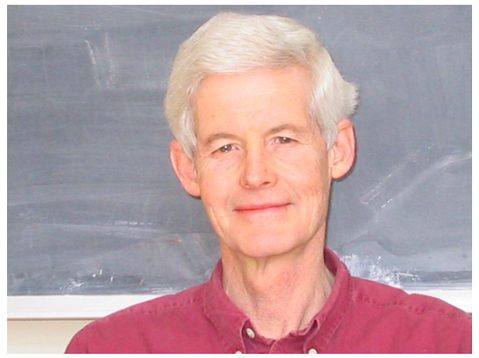


Why so much work on SAT

- Many interesting and computationally difficult problems can be reduced to SAT

- Boolean satisfiability is the first problem proved to be NP-complete

- Key idea: write one really good SAT solver and reduce all other NP-complete problems to SAT



Stephen Cook

Applications of SAT

- Motivation: Some products, such as cars, are highly customizable
- For example, Mercedes C class has a total of >650 options!
- Leather interior, seat heating, thermotronic comfort air conditioning, high-capacity battery, ventilated seats, heated steering wheel, 64-color LED ambient lighting, blind spot assist...

Lots of Options = Lots of dependencies

"Thermotronic comfort air conditioning requires high-capacity battery except when combined with gasoline engines of 3.2 liter capacity"

Maximum Satisfiability (MaxSAT)

- Problem: Given formula F in CNF, find assignment maximizing the number of satisfied clauses of F.

- If F is satisfiable, the solution to the MaxSAT problem is the number of clauses in F.

- If F is unsatisfiable, we want to find a maximum subset of F's clauses whose conjunction is satisfiable.

- What is a solution for the MaxSAT problem: $(a \vee b) \wedge \neg a \wedge \neg b$

Partial MaxSAT

- Similar to MaxSAT, but we distinguish between two kinds of clauses.
- Hard clauses: Clauses that must be satisfied
- Soft clauses: Clauses that we would like to, but do not have to satisfy
- Partial MaxSAT: Given CNF formula F where each clause is marked as hard or soft, find an assignment that satisfies all hard clauses and maximizes the number satisfied soft clauses

More on partial MaxSAT

- Both regular SAT and MaxSAT are special cases of partial MaxSAT
- In normal SAT, all clauses are hard clauses
- In MaxSAT, all clauses are implicitly soft clauses
- In this sense, Partial MaxSAT is a generalization over both SAT and MaxSAT

Partial weighted MaxSAT

- There is even one more generalization over Partial MaxSAT: Partial Weighted MaxSAT

- In addition to being hard and soft, clauses also have weights (e.g., indicating their importance)

- Partial Weighted MaxSAT problem: Find assignment maximizing the sum of weights of satisfied soft clauses

- Partial MaxSAT is an instance of partial weighted MaxSAT where all clauses have equal weight

Configuration Management

Given a configuration, consisting of a set of components, their dependencies, and conflicts:

- Decide if a new component can be added to the configuration.

- Add the component while optimizing some linear function.

- If the component cannot be added, find a way to add it by removing as few conflicting components from the current configuration as possible.

Configuration Management

Given a configuration, consisting of a set of components, their dependencies, and conflicts:

- Decide if a new component can be added to the configuration.

SAT

- Add the component while optimizing some linear function.

- If the component cannot be added, find a way to add it by removing as few conflicting components from the current configuration as possible.

Configuration Management

Given a configuration, consisting of a set of components, their dependencies, and conflicts:

- Decide if a new component can be added to the configuration.

SAT

- Add the component while optimizing some linear function.

Pseudo-Boolean Constraints

- If the component cannot be added, find a way to add it by removing as few conflicting components from the current configuration as possible.

Configuration Management

Given a configuration, consisting of a set of components, their dependencies, and conflicts:

- Decide if a new component can be added to the configuration.

SAT

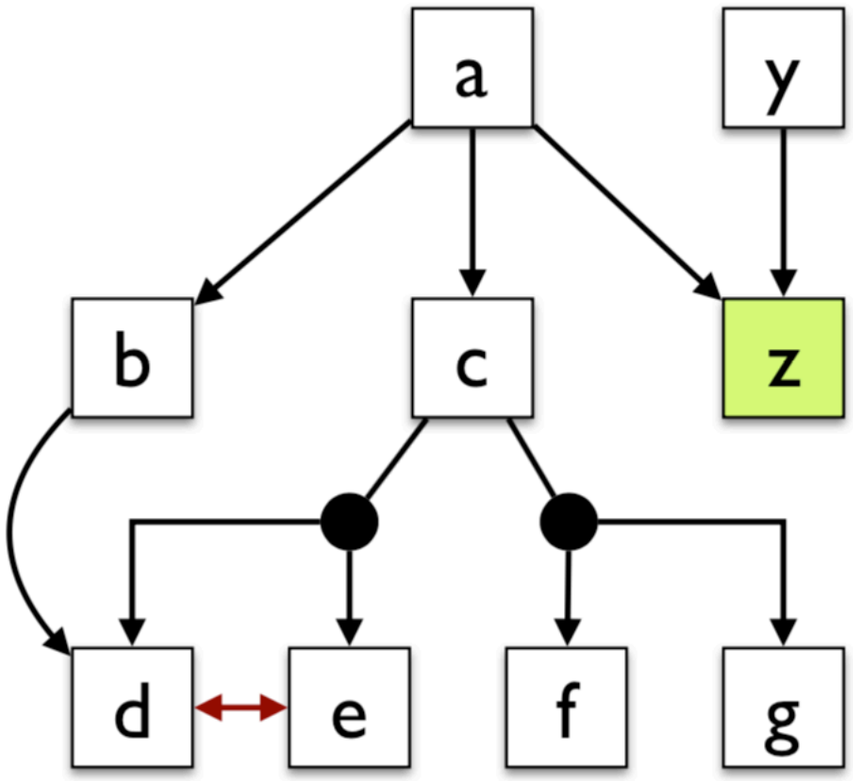
- Add the component while optimizing some linear function.

Pseudo-Boolean Constraints

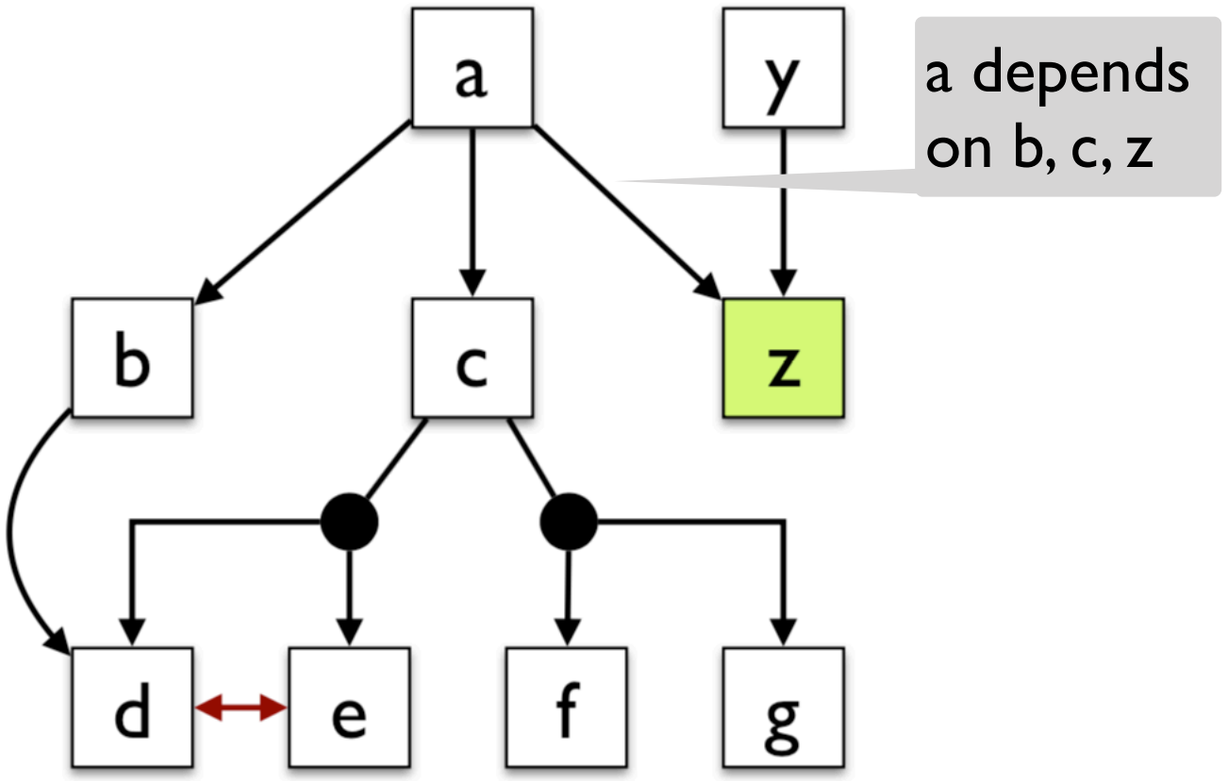
- If the component cannot be added, find a way to add it by removing as few conflicting components from the current configuration as possible.

Partial (Weighted) MaxSAT

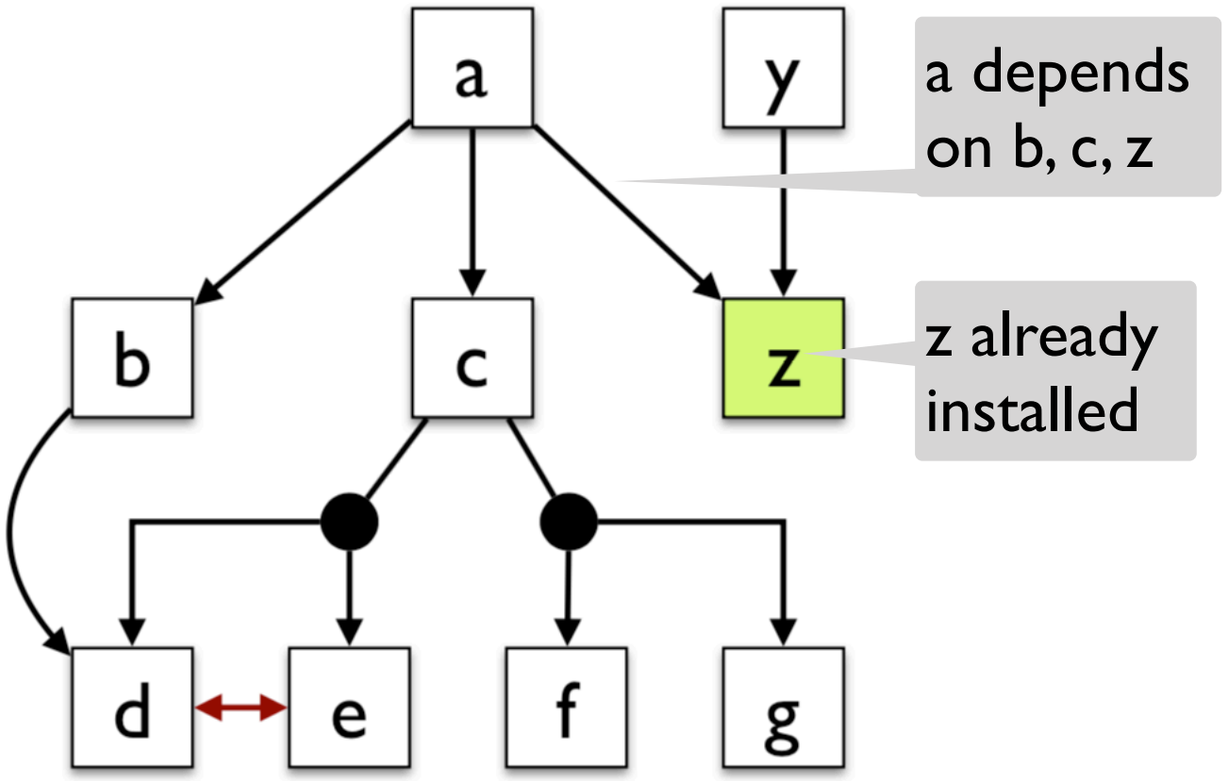
SAT for component installation



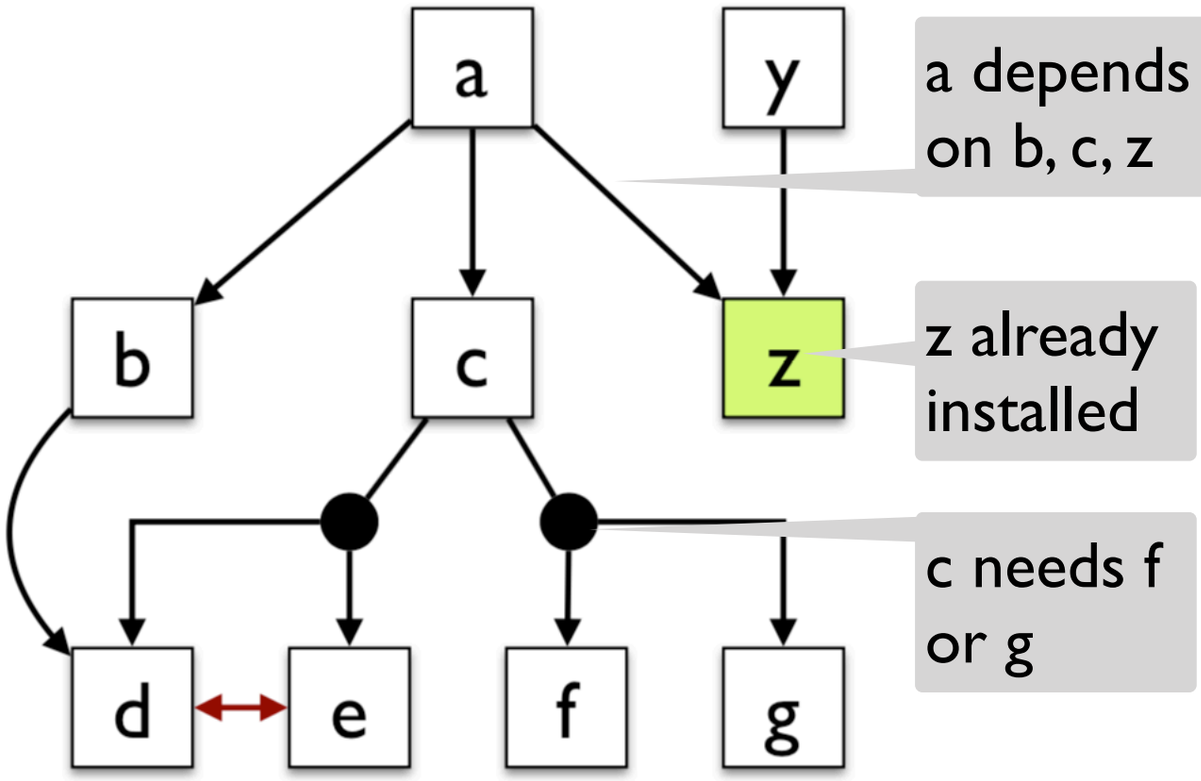
SAT for component installation



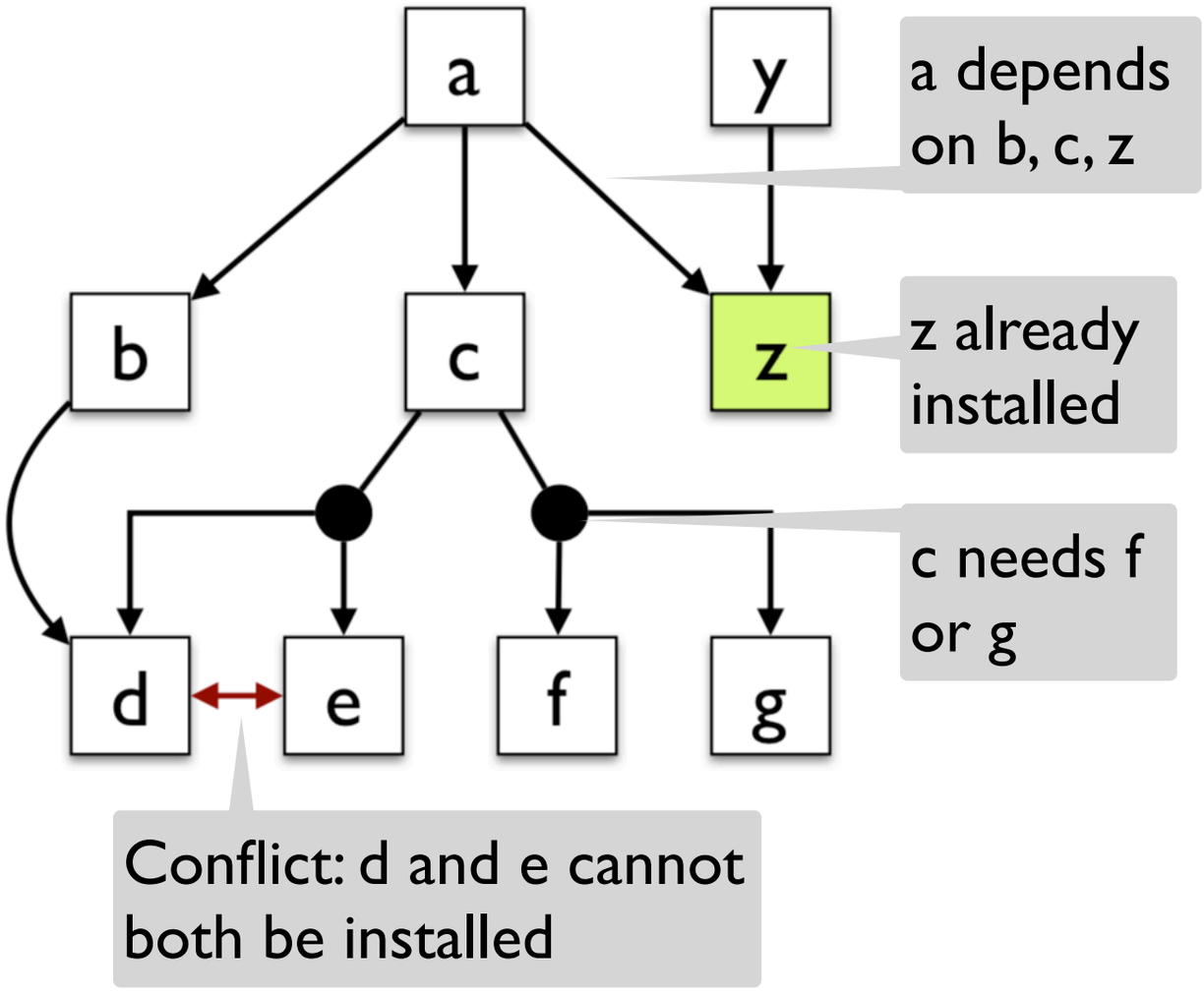
SAT for component installation



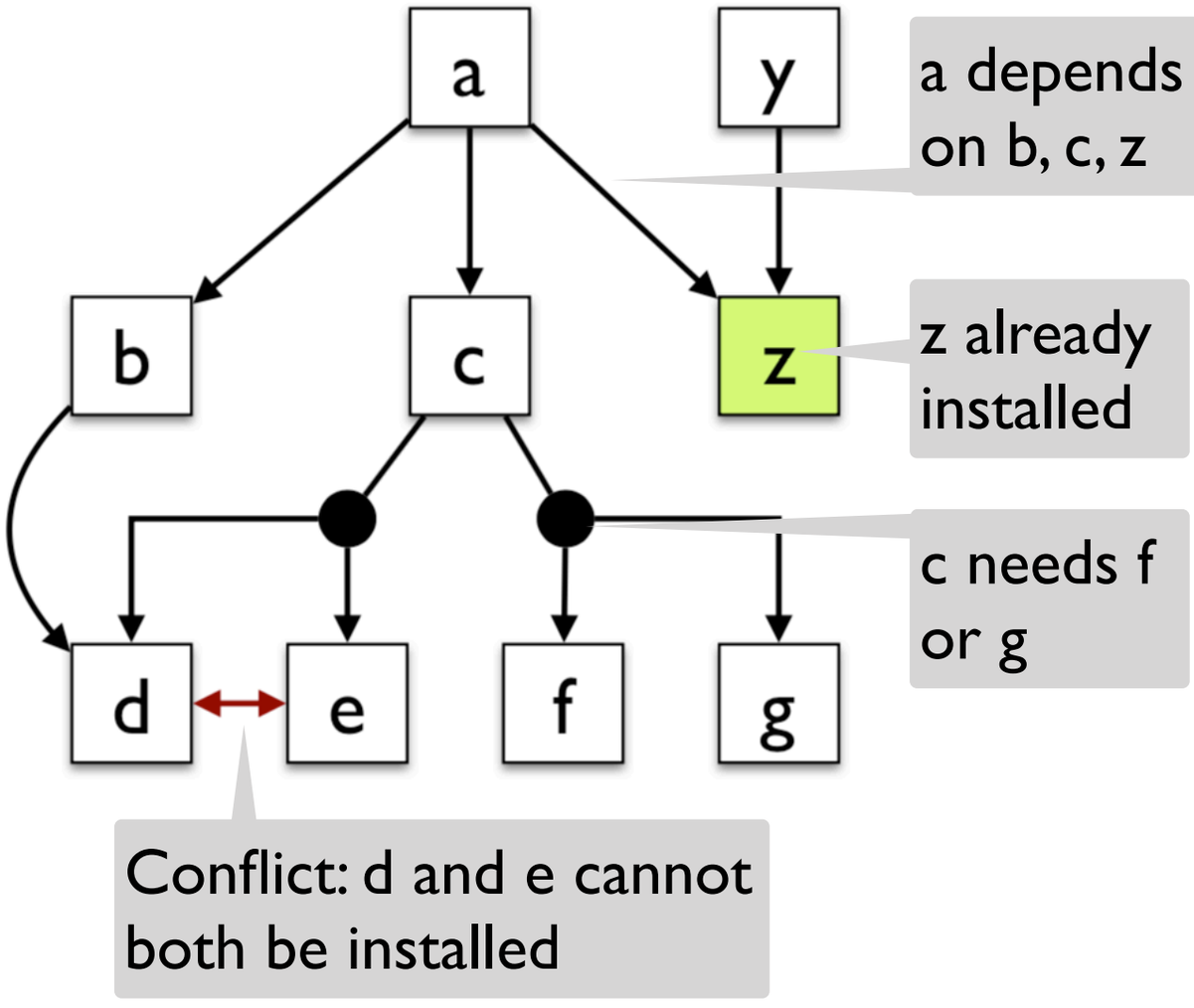
SAT for component installation



SAT for component installation

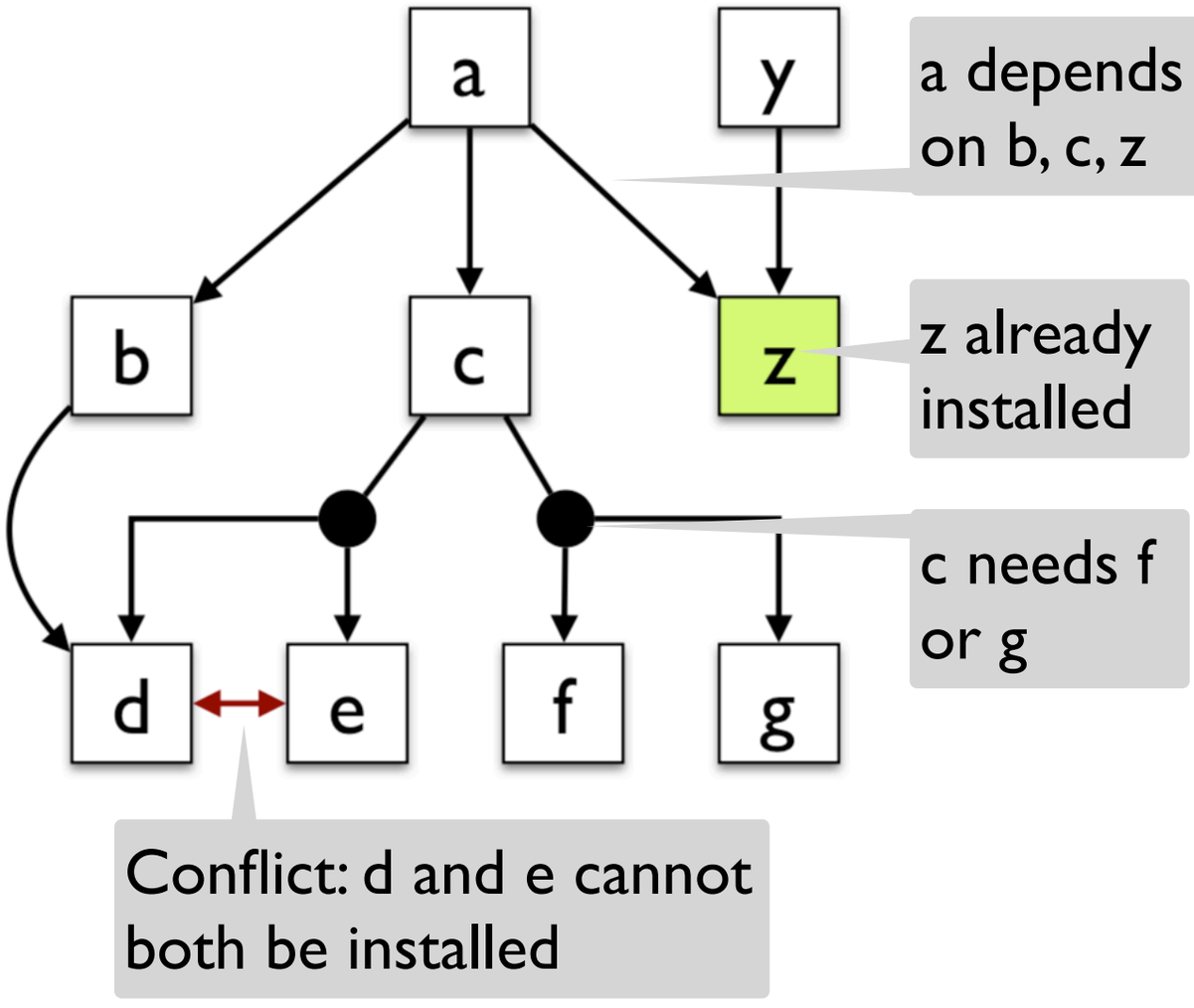


SAT for component installation



To install a, CNF constraints are:

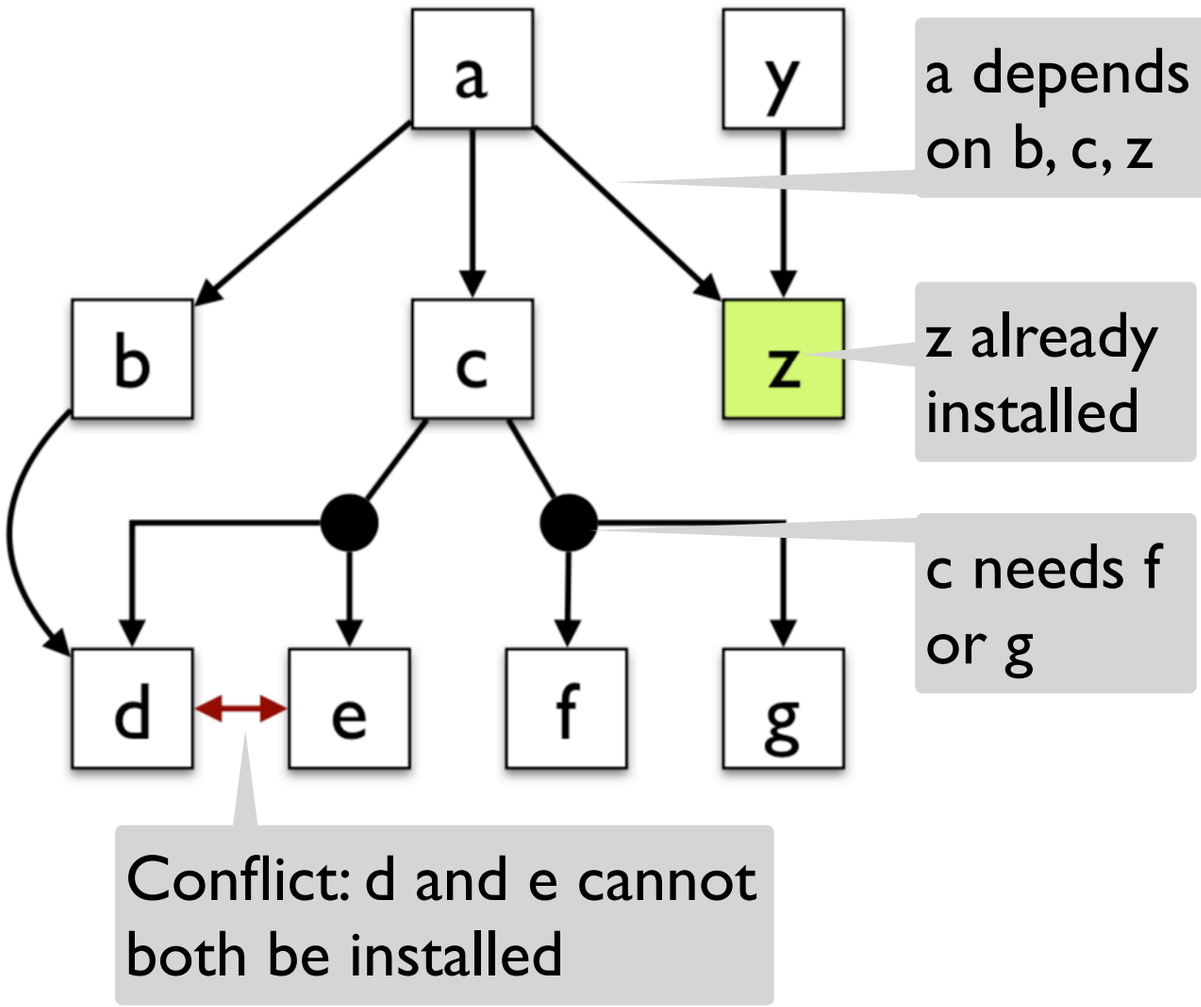
SAT for component installation



To install a, CNF constraints are:

$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$

SAT for component installation



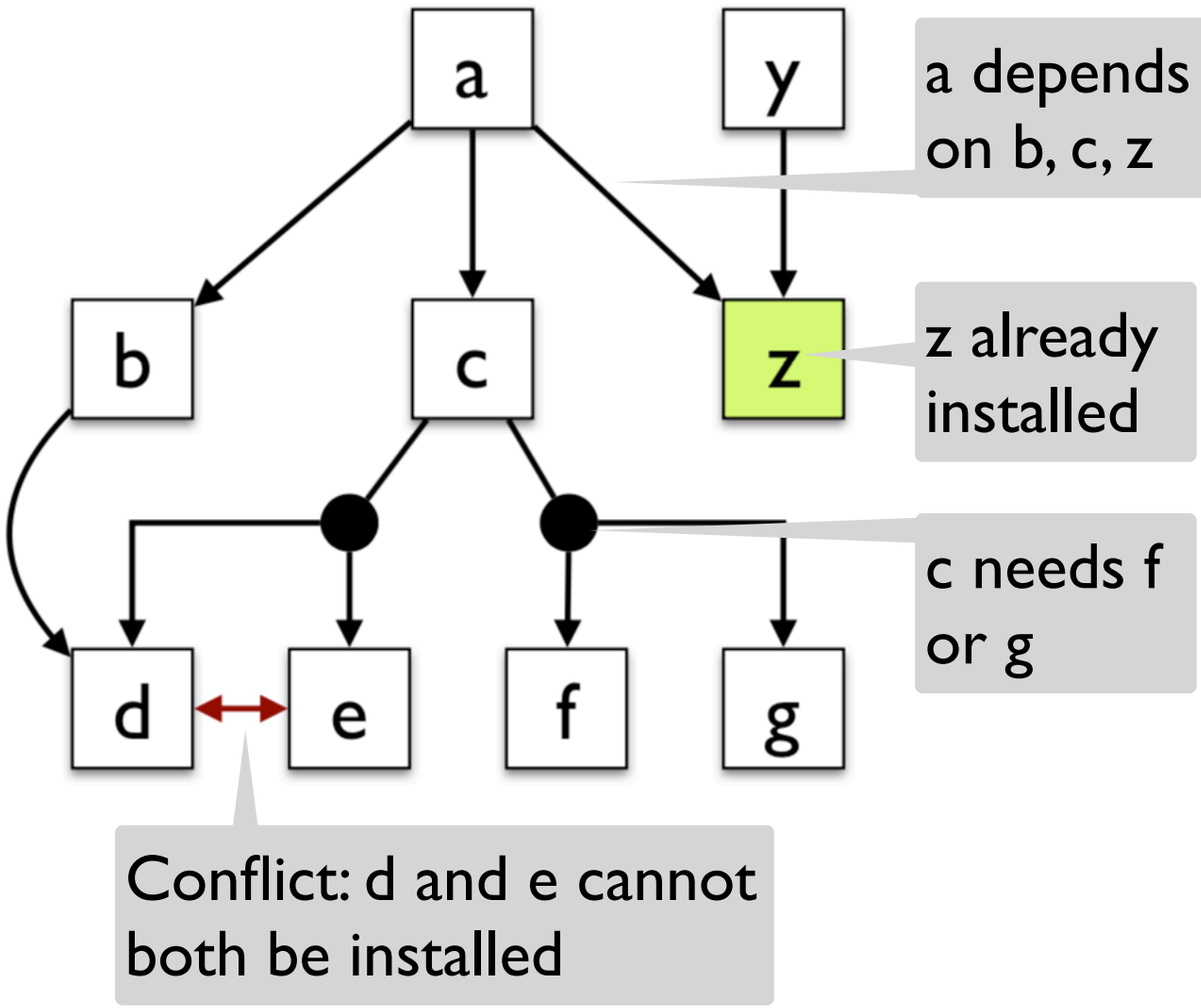
To install a, CNF constraints are:

$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$

$(\neg b \vee d) \wedge$

$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$

SAT for component installation



To install a, CNF constraints are:

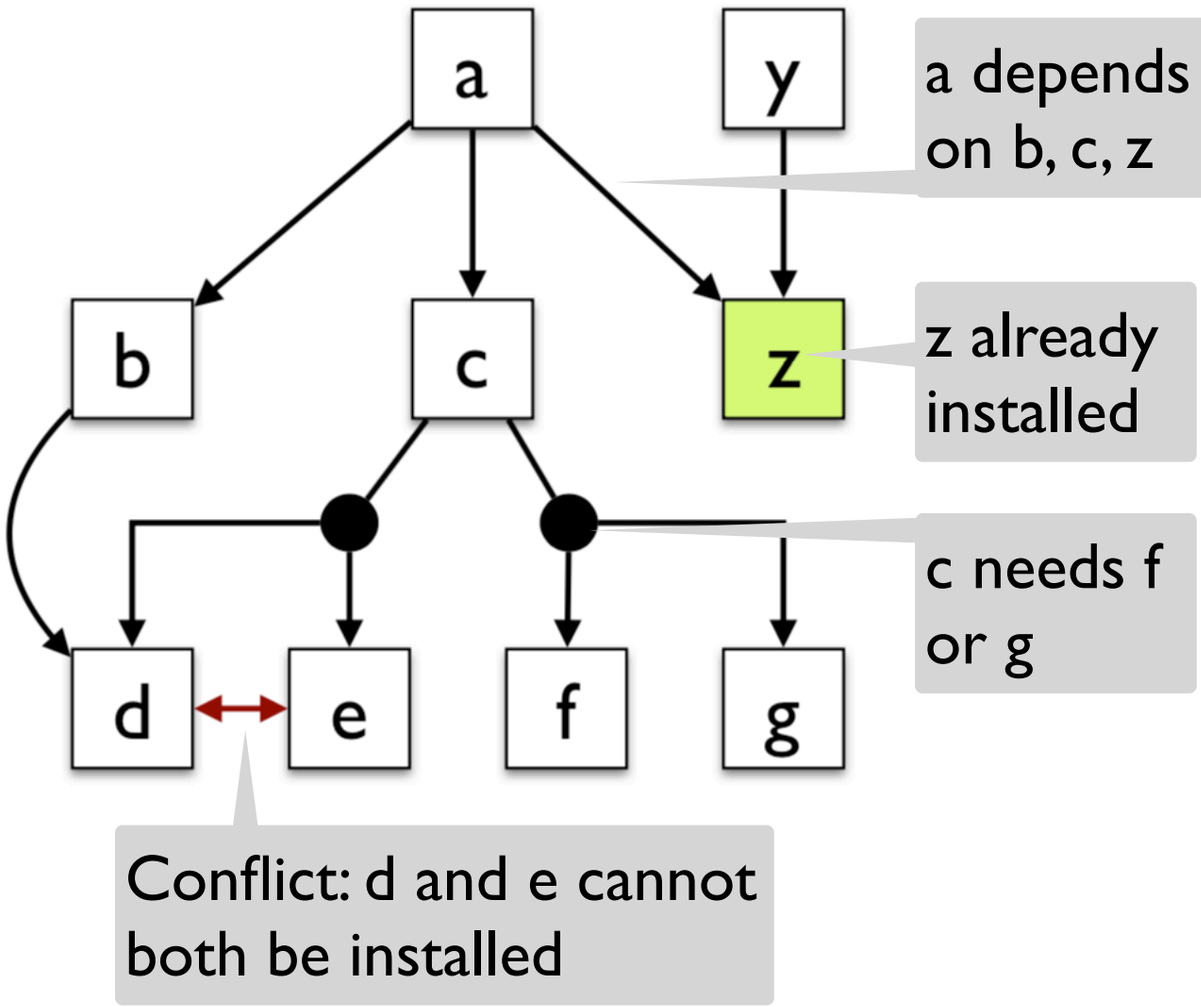
$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$

$(\neg b \vee d) \wedge$

$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$

$(\neg d \vee \neg e) \wedge$

SAT for component installation



To install a, CNF constraints are:

$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$

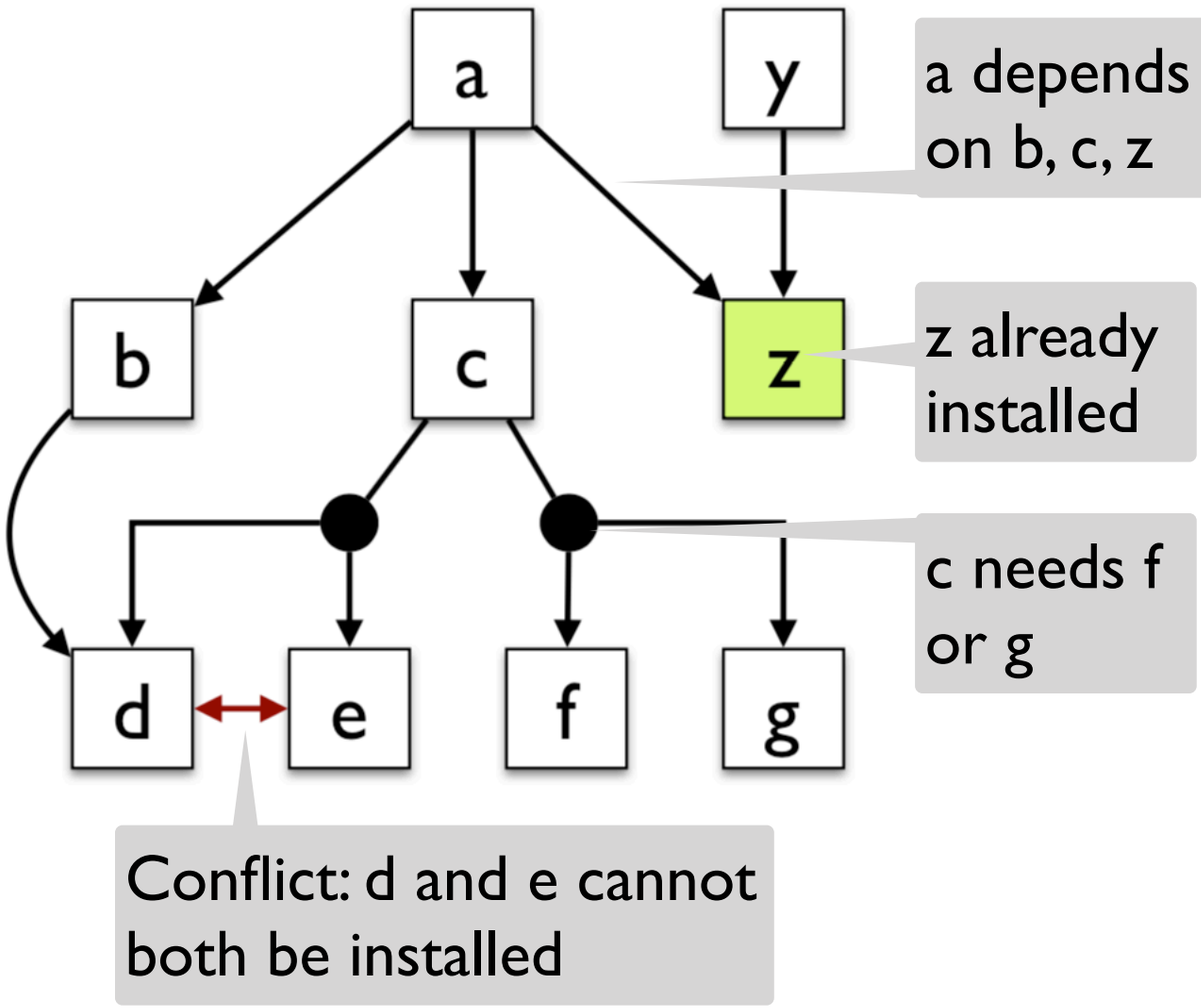
$(\neg b \vee d) \wedge$

$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$

$(\neg d \vee \neg e) \wedge$

$(\neg y \vee z) \wedge$

SAT for component installation



To install a, CNF constraints are:

$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$

$(\neg b \vee d) \wedge$

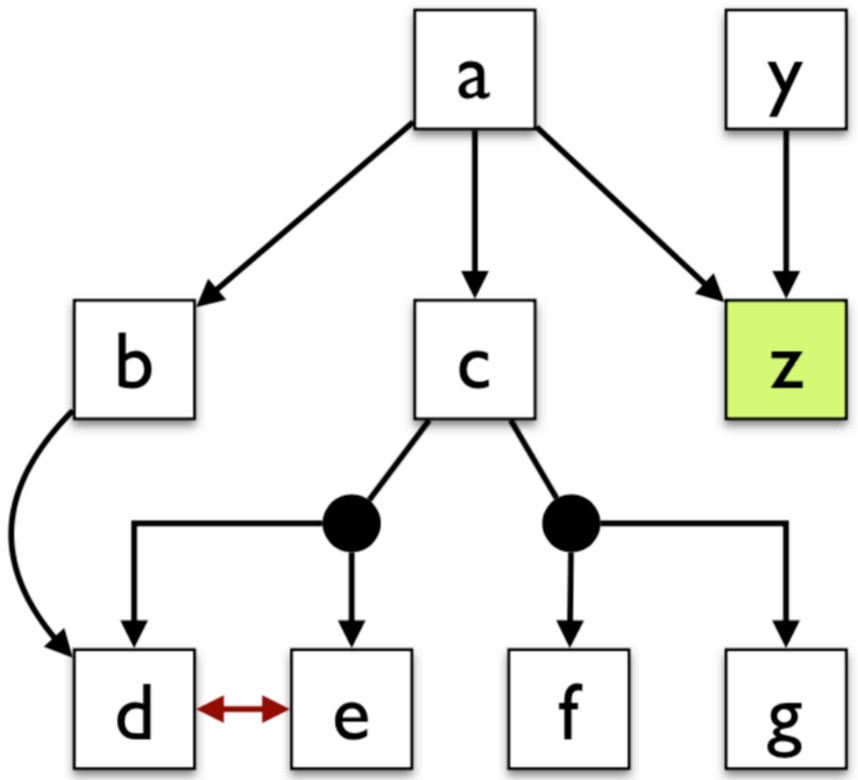
$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$

$(\neg d \vee \neg e) \wedge$

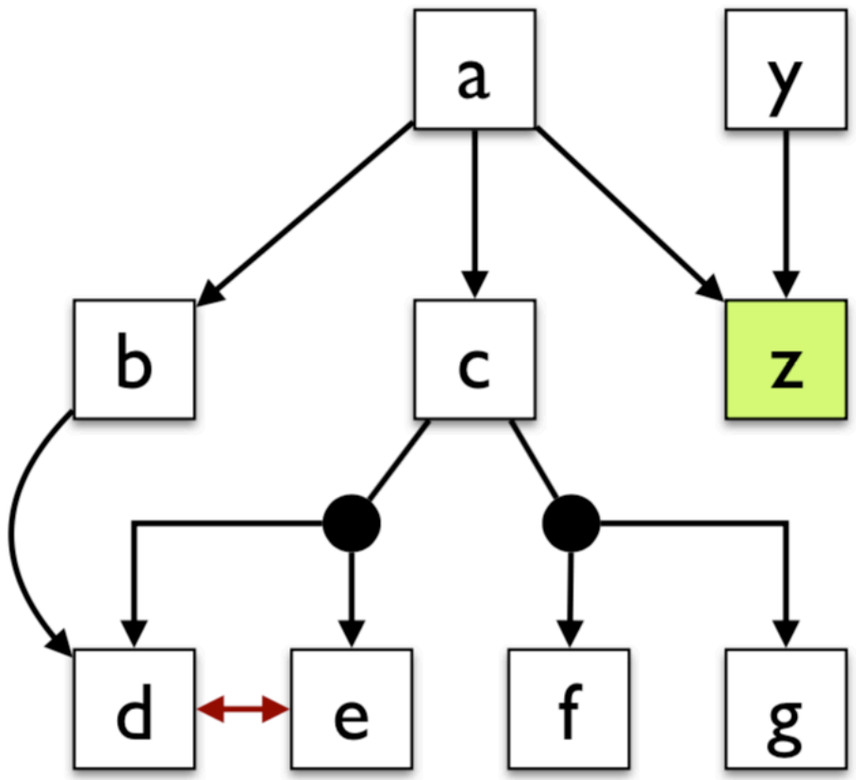
$(\neg y \vee z) \wedge$

$a \wedge z$

Optimal installation

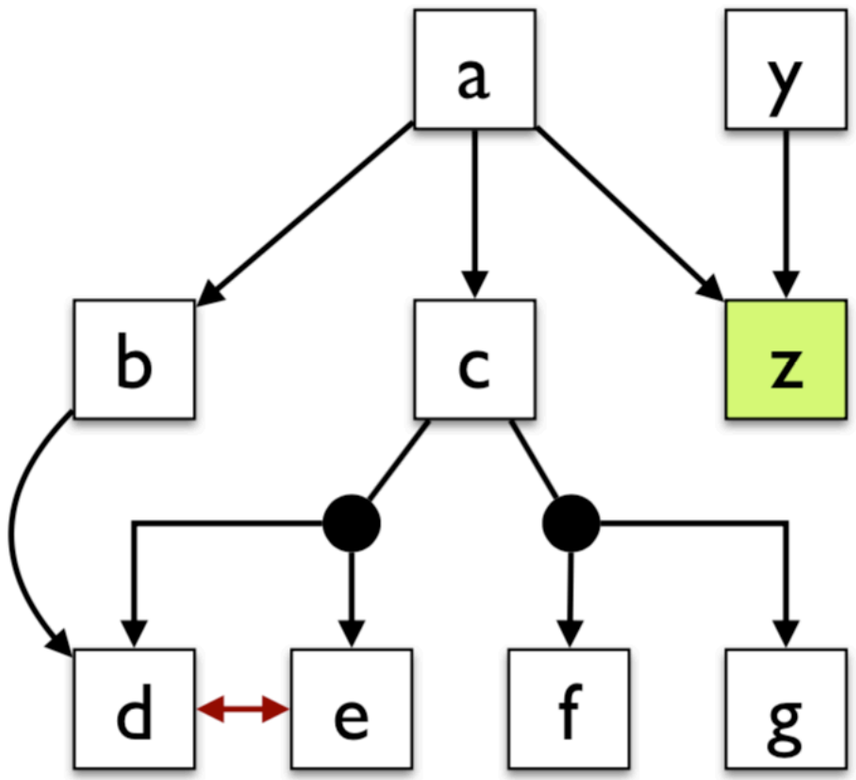


Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

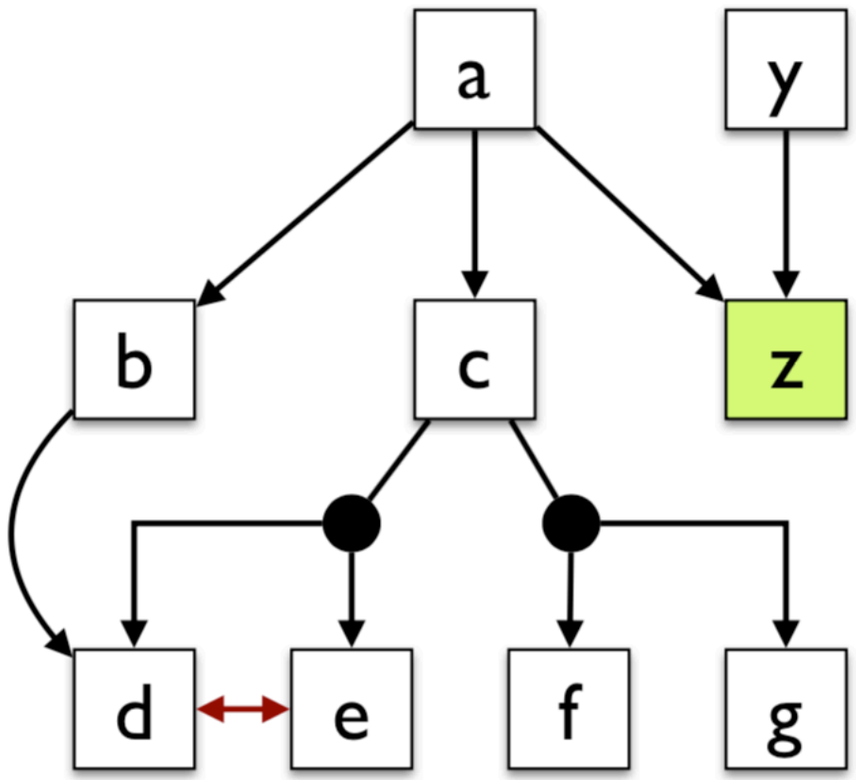
Optimal installation



Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Optimal installation



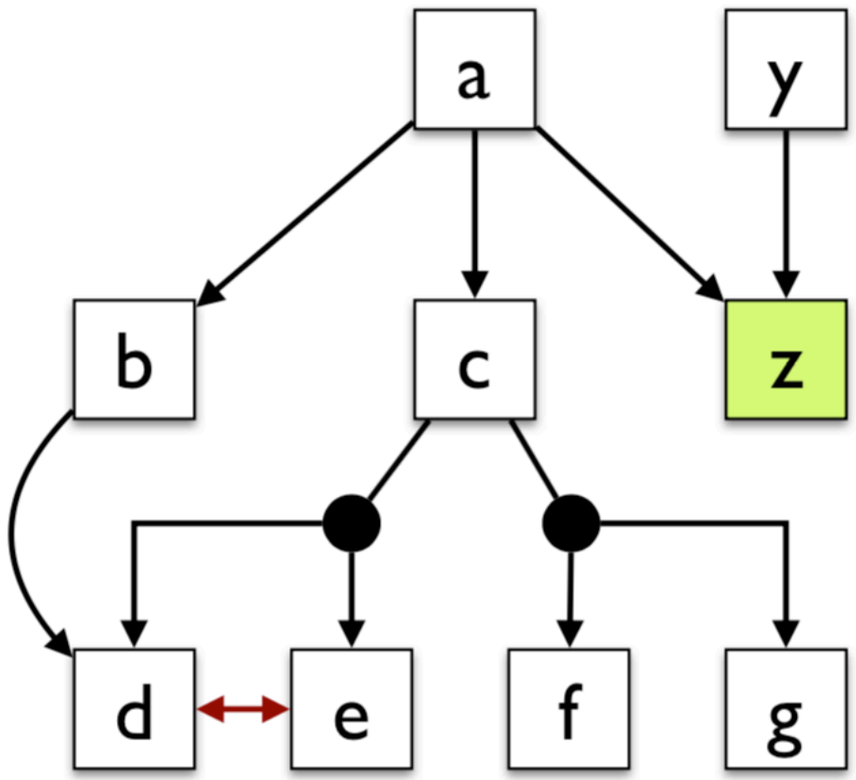
Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min $c_1x_1 + ... + c_nx_n$

$a_{11}x_1 + ... + a_{1n}x_n \geq b_1 \wedge ... \wedge$

Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

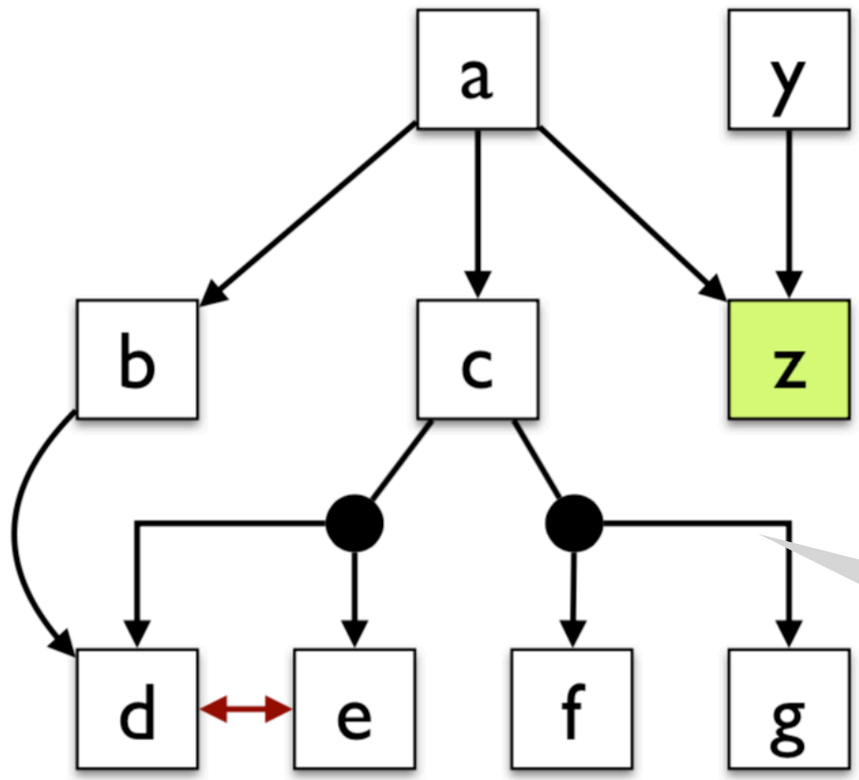
Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min $c_1x_1 + \ldots + c_nx_n$

$a_{11}x_1 + \ldots + a_{1n}x_n \geq b_1 \wedge \ldots \wedge$

$a_{k1}x_1 + \ldots + a_{kn}x_n \geq b_k$

Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

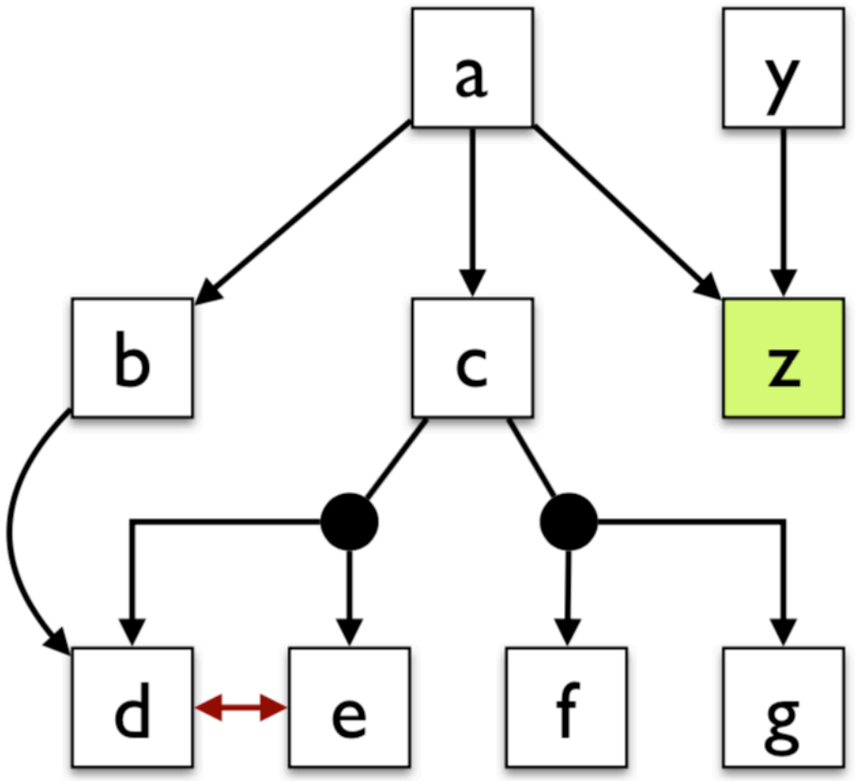
min $c_1x_1 + ... + c_nx_n$

$a_{11}x_1 + ... + a_{1n}x_n \geq b_1 \wedge ... \wedge$

$a_{k1}x_1 + ... + a_{kn}x_n \geq b_k$

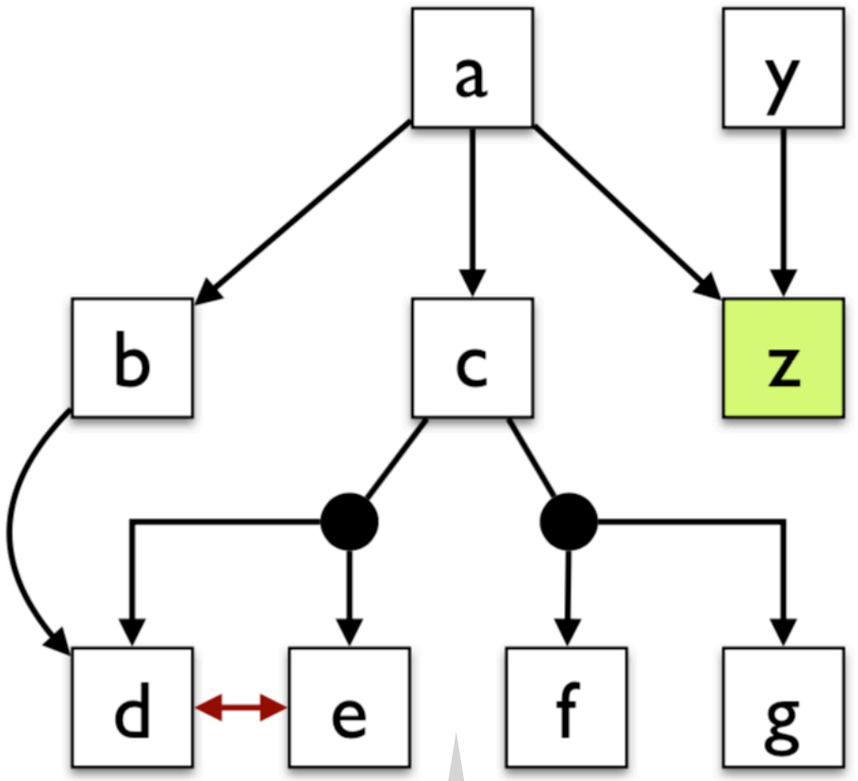
$(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge$
$(\neg b \vee d) \wedge$
$(\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge$
$(\neg d \vee \neg e) \wedge$
$(\neg y \vee z) \wedge$
$a \wedge z$

Optimal installation



0-1 ILP

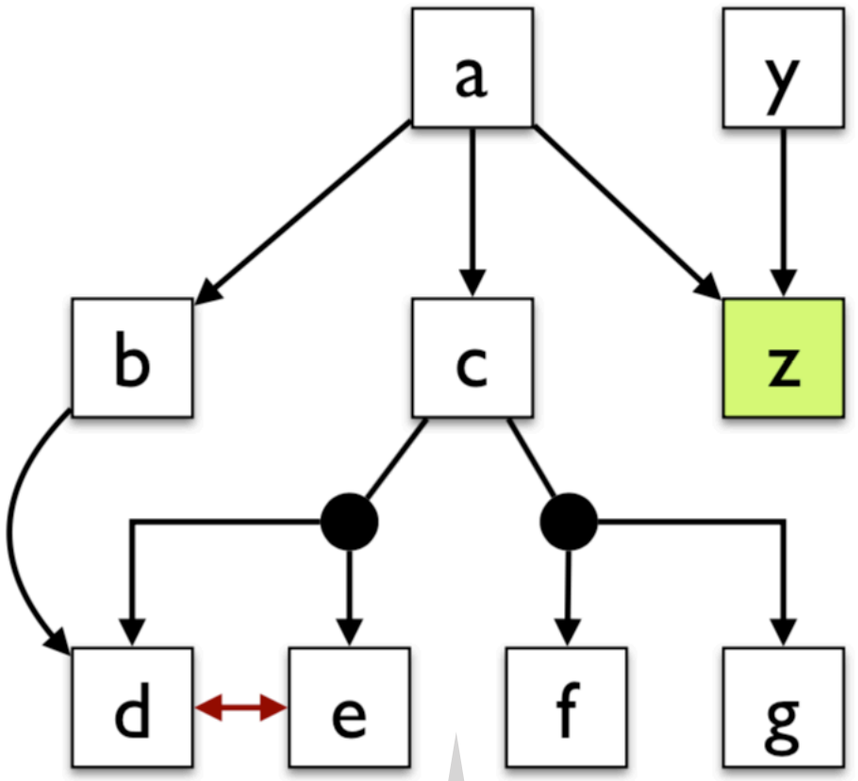
Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

0-1 ILP

Optimal installation

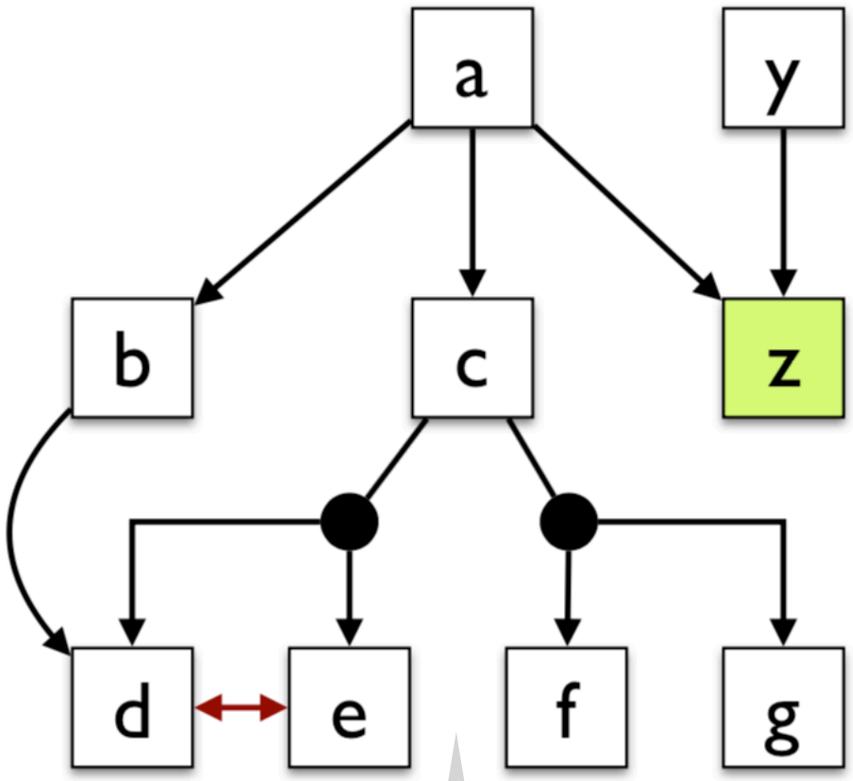


Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

0-1 ILP

Optimal installation



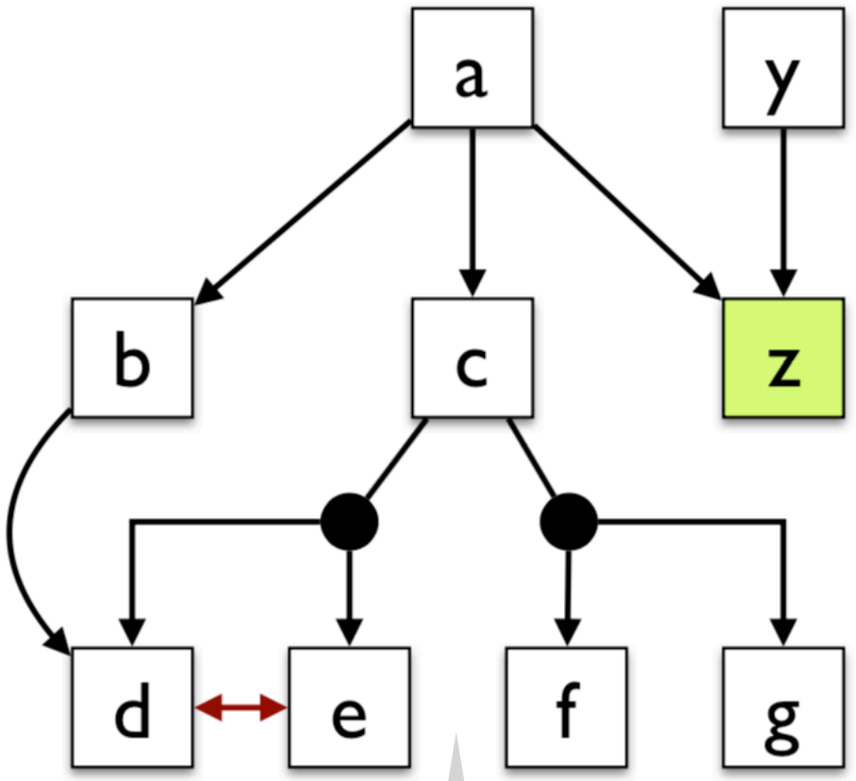
Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min a+b+c+d+e+5f +2g+y+0z

0-1 ILP

Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

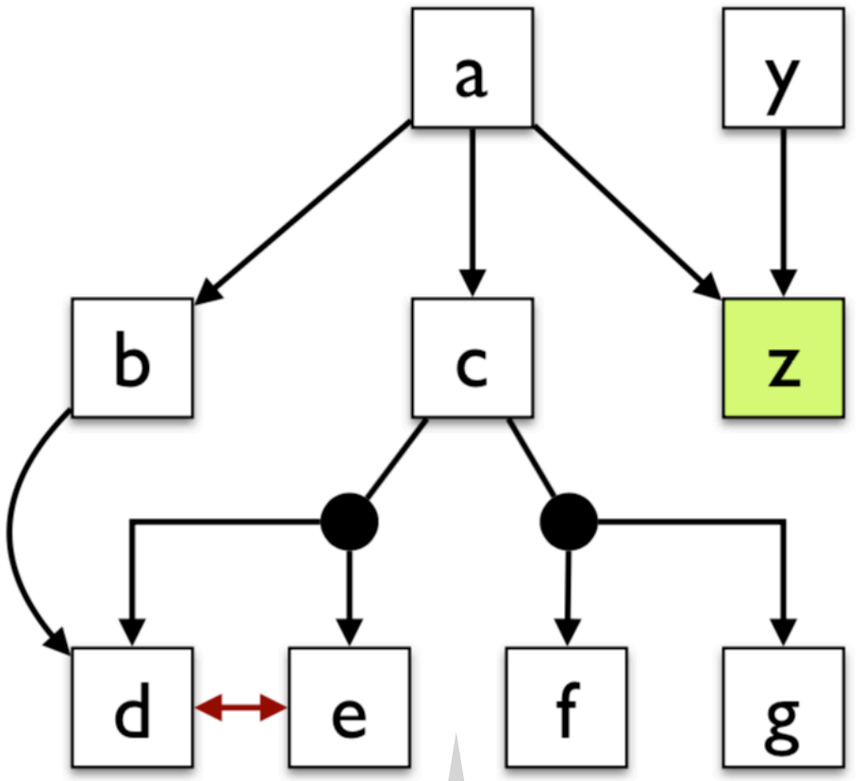
Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min a+b+c+d+e+5f +2g+y+0z

$(-a + b \geq 0) \wedge (-a + c \geq 0) \wedge (-a + z \geq 0) \wedge$

0-1 ILP

Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min a+b+c+d+e+5f +2g+y+0z

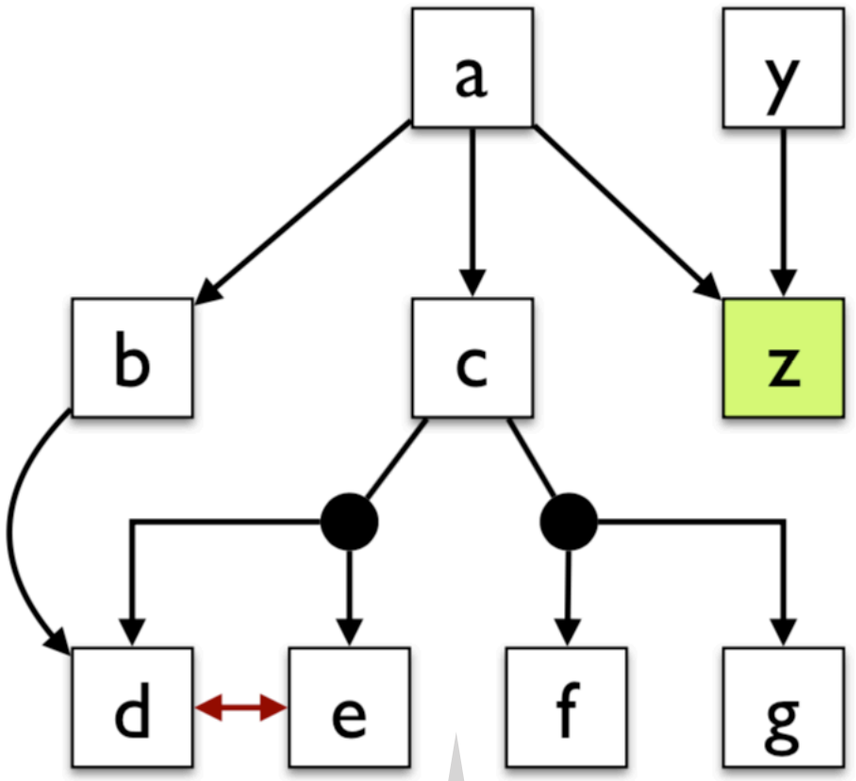
$(-a + b \geq 0) \wedge (-a + c \geq 0) \wedge (-a + z \geq 0) \wedge$

$(-b + d \geq 0) \wedge$

$(-c + d + e \geq 0) \wedge (-c + f + g \geq 0) \wedge$

0-1 ILP

Optimal installation



Assume f and g are 5MB and 2MB each, and all other components are 1MB. How to install a, while minimizing total size?

Pseudo-boolean solvers accept a linear function to minimize, in addition to a (weighted) CNF.

min a+b+c+d+e+5f +2g+y+0z

$(-a + b \geq 0) \wedge (-a + c \geq 0) \wedge (-a + z \geq 0) \wedge$

$(-b + d \geq 0) \wedge$

$(-c + d + e \geq 0) \wedge (-c + f + g \geq 0) \wedge$

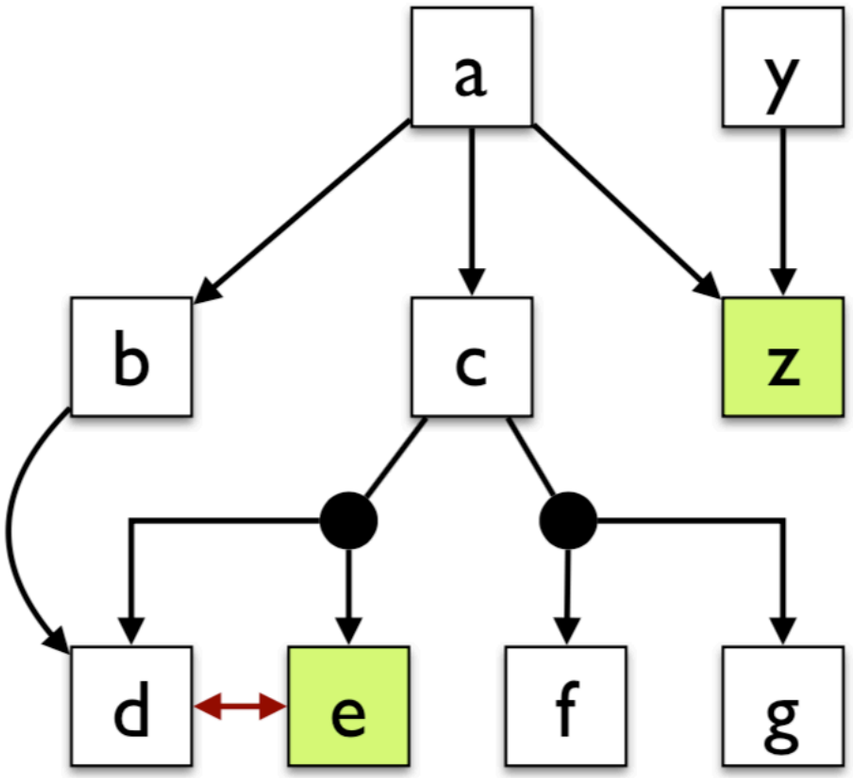
$(-d + -e \geq -1) \wedge$

$(-y + z \geq 0) \wedge$

$(a \geq 1) \wedge (z \geq 1)$

0-1 ILP

Installation with conflicts



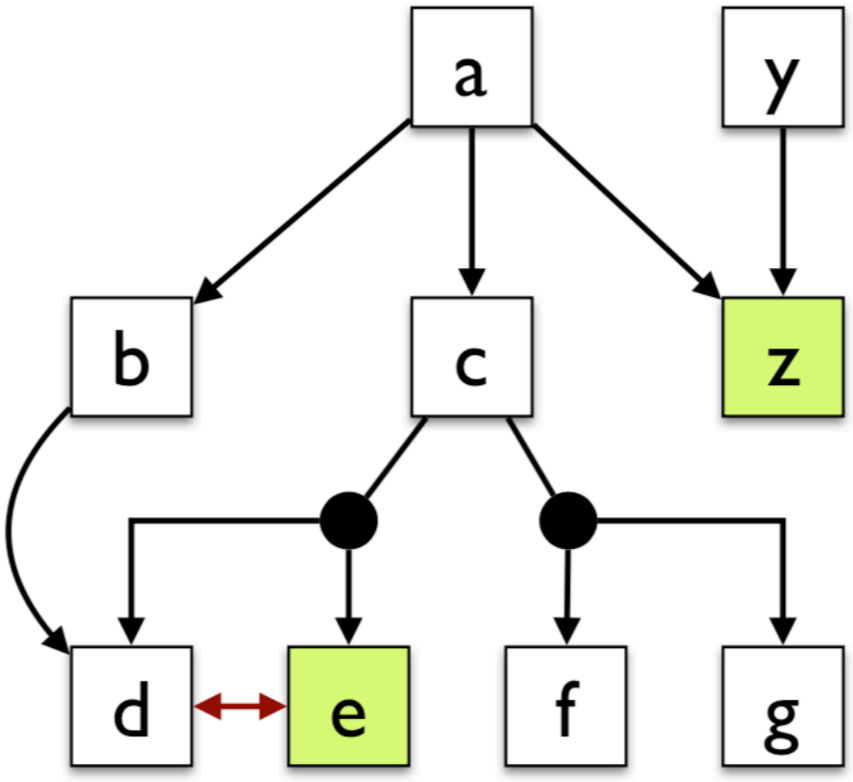
Partial MaxSAT solver takes as input a set of **hard** clauses and a set of **soft** clauses, and it produces an assignment that satisfies all hard clauses and the greatest number of soft clauses.

To install a, while minimizing the number of removed components, Partial MaxSAT constraints are:

hard: $(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge (\neg b \vee d) \wedge (\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge (\neg d \vee \neg e) \wedge (\neg y \vee z) \wedge a$

soft: $e \wedge z$

Installation with conflicts



Partial MaxSAT solver takes as input a set of **hard** clauses and a set of **soft** clauses, and it produces an assignment that satisfies all hard clauses and the greatest number of soft clauses.

To install a, while minimizing the number of removed components, Partial MaxSAT constraints are:

hard: $(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee z) \wedge (\neg b \vee d) \wedge (\neg c \vee d \vee e) \wedge (\neg c \vee f \vee g) \wedge (\neg d \vee \neg e) \wedge (\neg y \vee z) \wedge a$

soft: $e \wedge z$

A funny story of my friends

Lisbon Wedding: Seating arrangements using MaxSAT

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

Justine Sherry
Carnegie Mellon University
sherry@cs.cmu.edu

***Abstract*—Having a perfect seating arrangement for weddings is not an easy task. Can Alice sit next to Bob? Can we ensure that Charles and his ex-girlfriend Eve not be seated together? Meeting such constraints is classically one of the most difficult tasks in planning a wedding – and guests will not accept 'it's NP-complete!' as an excuse for poor seating arrangements. We discuss how MaxSAT can provide the optimal seating arrangement for a perfect wedding, saving brides and grooms (including the authors) from hours of struggle.**

I. Introduction

This benchmark description describes the encoding used for the wedding seating arrangement for our wedding in Lisbon. We needed to seat our guests according to a long list of constraints. For example, members of the same family should sit together; friends who went to school together should sit together; individuals with a history of conflict should be seated apart; etc. We wanted to maximize the happiness of our guests and what better way to do that than to encode the problem into MaxSAT! MaxSAT was an ideal solution for our own wedding: i) it saved us tens of hours, ii) it was stress free, and iii) in the rare case that a guest complained about their seating arrangement, we just blamed the algorithm![1]

II. MaxSAT Encoding

When making a seating arrangement, we first need to define the size of each table and how many guest we have. Assume that our guests are defined by the set P and the tables are defined by the set T. Each table has at least l guests and at most u guests.

$$\forall_{p \in P} \sum_{t \in T} p_t = 1$$

- Each table will have at most u guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \leq u$$

- Each table will have at least l guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \geq l$$

Since some guests may have disagreements with each other, we also included some exclusion constraints that guarantee that guests which have conflicts with each other are not seated in the same table. For every pair of guests p and p' that have a conflict with each other we include the following constraints that guarantee that they will not seat together:

$$\forall_{t \in T} (p_t + p'_t \leq 1)$$

To enforce that if a person p is seated at table t then t will contain all labels belonging to p we add additional hard constraints that enforce that table t will contain all the labels from guests that are seated there:

$$\forall_{t \in T} \forall_{p \in P} \forall_{s \in S_p} (p_t \implies s_t^p)$$

Soft constraints. The soft constraints describe the common-

TODOs by next lecture

- The 3rd reading assignment is out
- Discuss your final project during office hour (last chance)