Automatic Boosting of Cross-Product Coverage Using Bayesian Networks

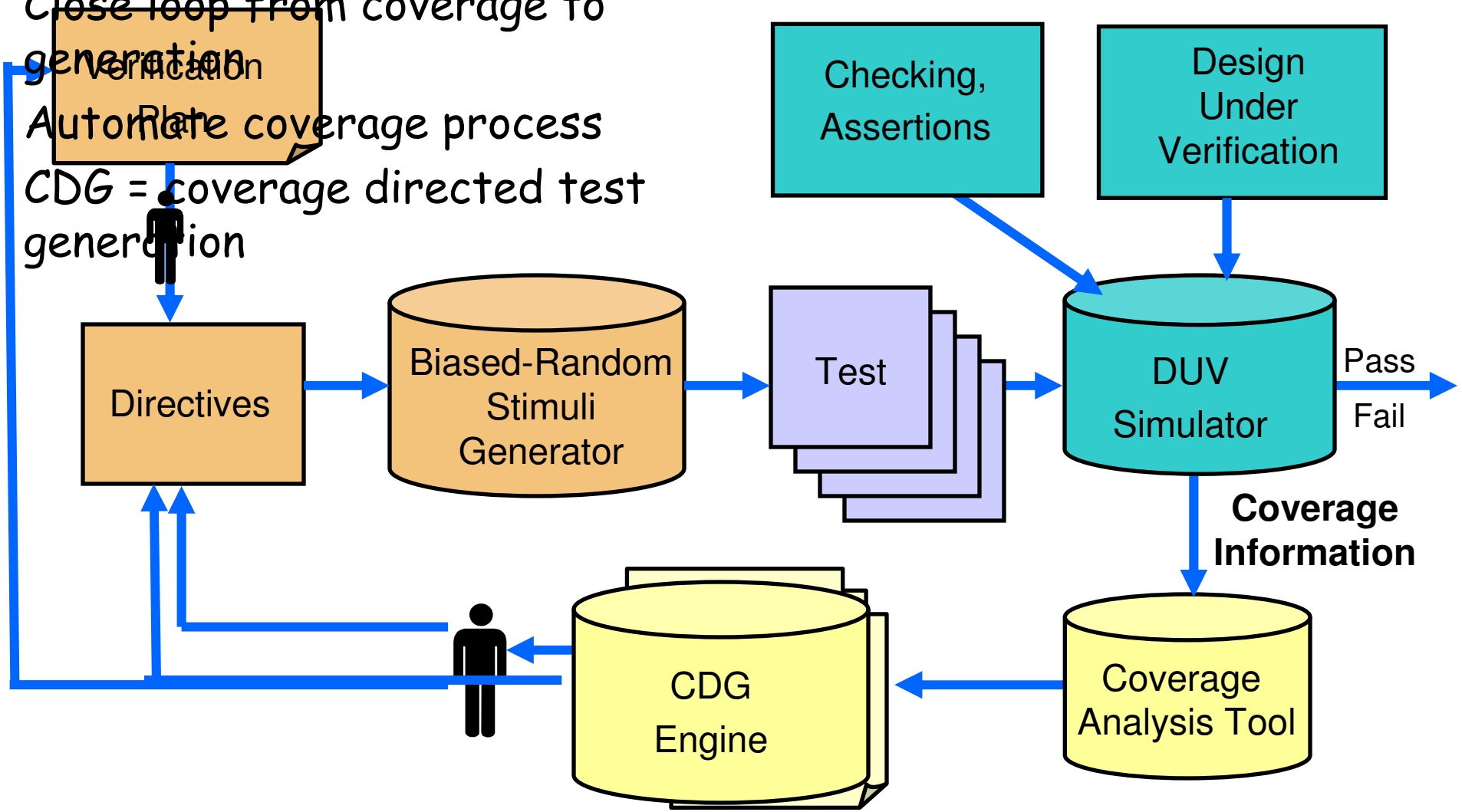
Dorit Baras, Laurent Fournier and Avi Ziv
Verification and Analytics Department,
IBM Haifa Research Lab

Outline

- Simulation Based Verification Environment
- The Coverage Booster
- CDG + Bayesian Networks
- Booster Structure
  - Feature Selection
  - Structure Learning
  - Parameter Learning
- Experiments
- Conclusions and Future Work

Simulation Based Verification Environment

- Close loop from coverage to generation
- Automate coverage process
- CDG = coverage directed test generation



Machine Learning Based CDG Engine

- Represents relations between
  - Directives (input)
  - Coverage Data (output)
- Uses examples to learn relations
  - Pairs of input-output. Also known as training set
- After learning
  - The engine is presented with queries
    - What directives to use in order to hit unseen coverage data
- At the heart of the CDG engine – **Bayesian Network**

Characteristics of Current CDG Process

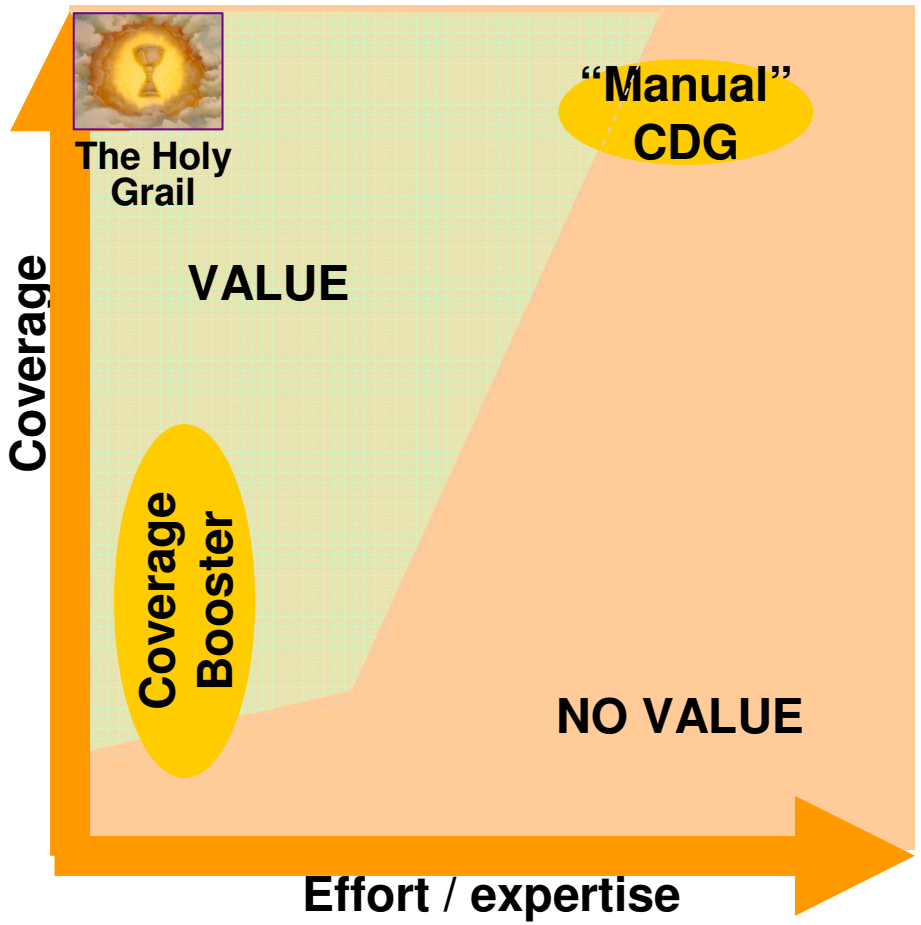
Beneficial

- Should reach 100% coverage
- Ability to hit hard-to-reach events

But…

- Integrated in late stages of design – provides less impact
- Expensive
  - Large amount of manual work
  - Domain knowledge
  - Expertise in machine learning techniques

- New approach: coverage booster



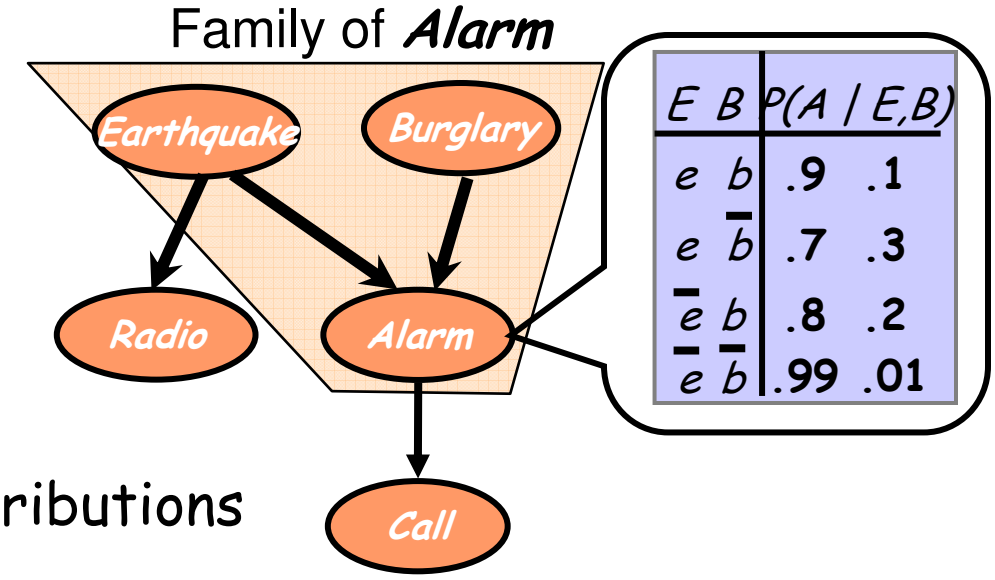
Bayesian Networks

- Compact representation of probability distributions via conditional independence
- Qualitative part:

  Directed acyclic graph (DAG)
  - Nodes - random variables
  - Edges - direct influence
- Quantitative part:

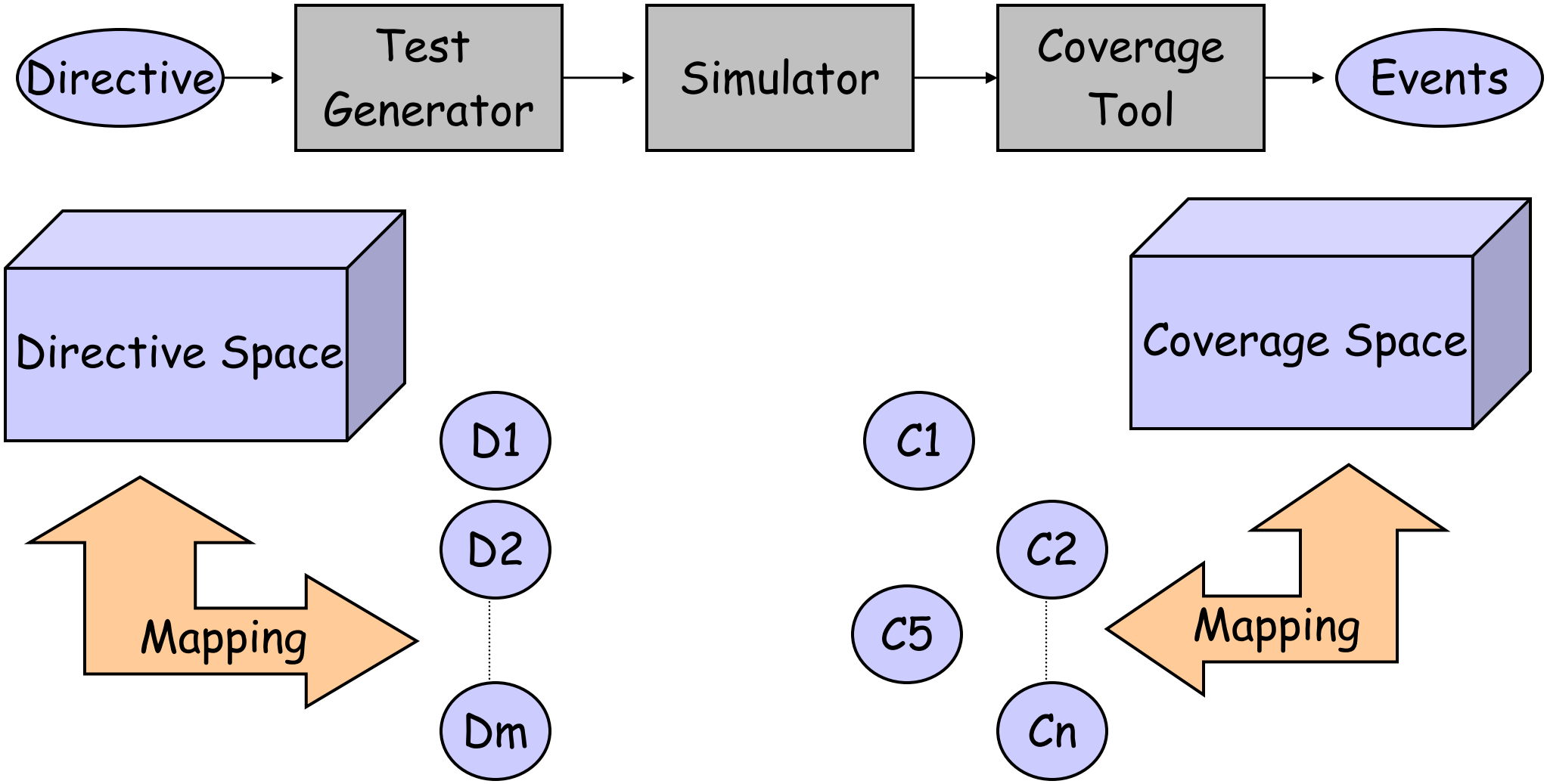
  Set of conditional probability distributions
- Together:

  Define a unique distribution in a factored form

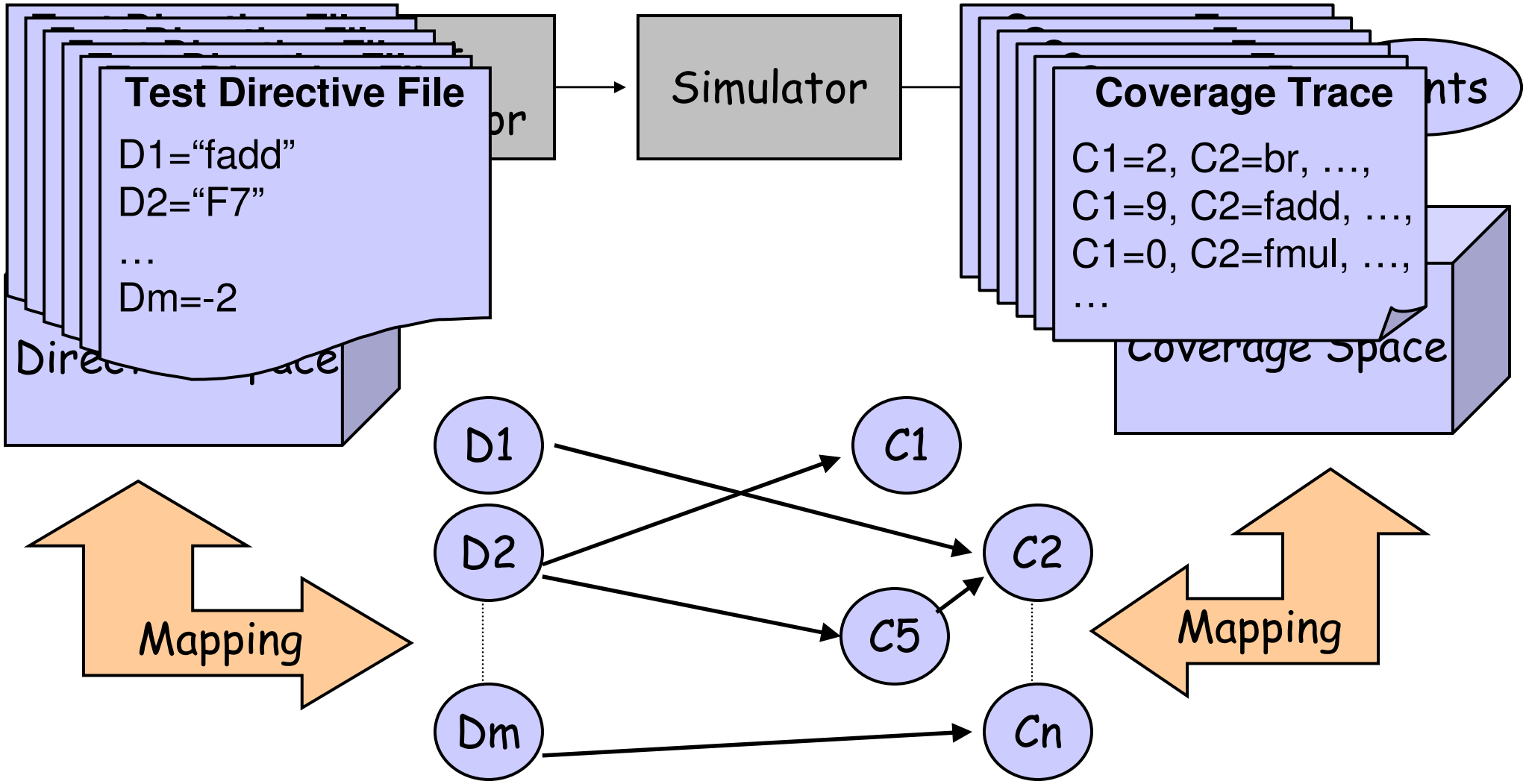


$$P(B,E,A,C,R) = P(B)P(E)P(A \mid B,E)P(R \mid E)P(C \mid A)$$

Bayesian Networks for CDG



Bayesian Networks for CDG

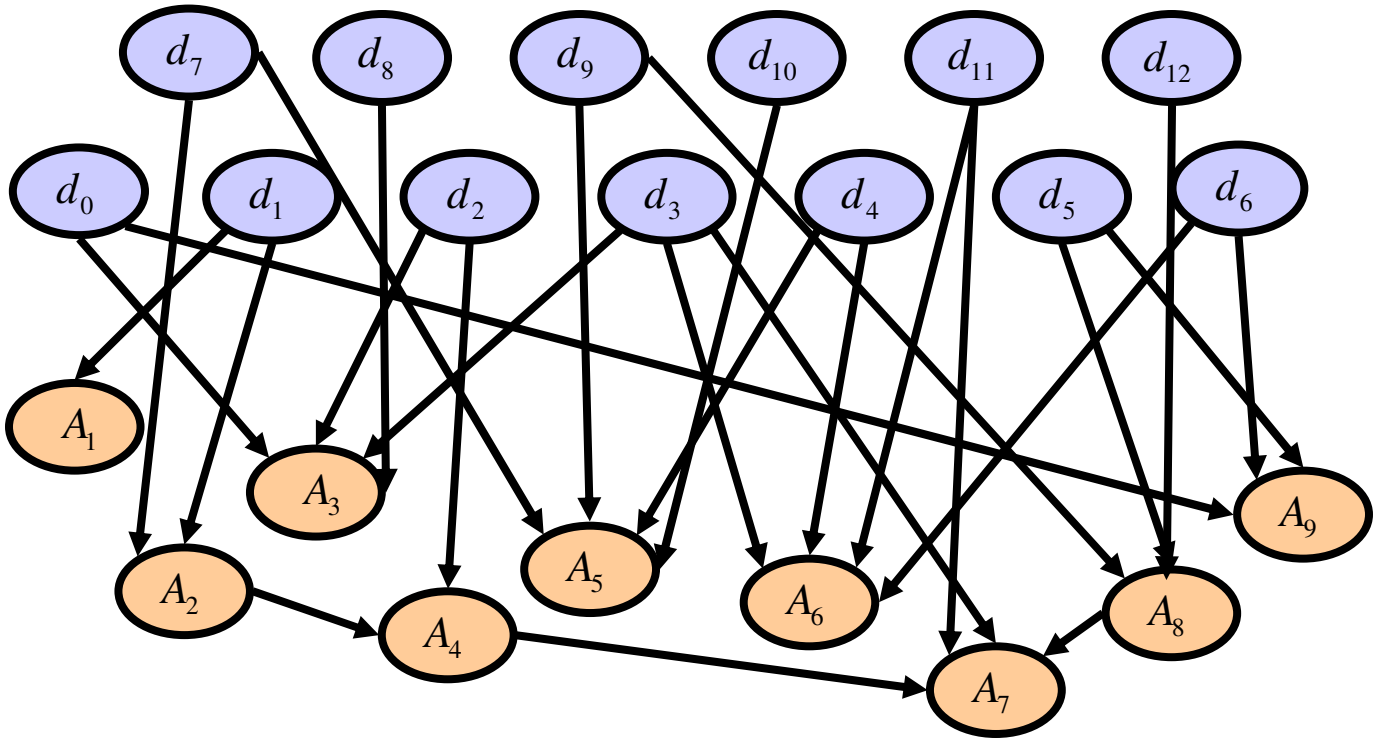


Learn conditional distributions: $P(C1 \mid D2), P(C2 \mid D1, C5), \ldots$

The Coverage Booster

- Goal: Design a fully automatic process for constructing CDG engines
- Benefits:
  - Reduce human effort to minimal
  - Provide assistance at early stages
- CDG engine
  - Construction of Bayesian networks
    - Structure
    - Conditional probability tables
  - Ability to answer queries
    - Generating directives to hit events

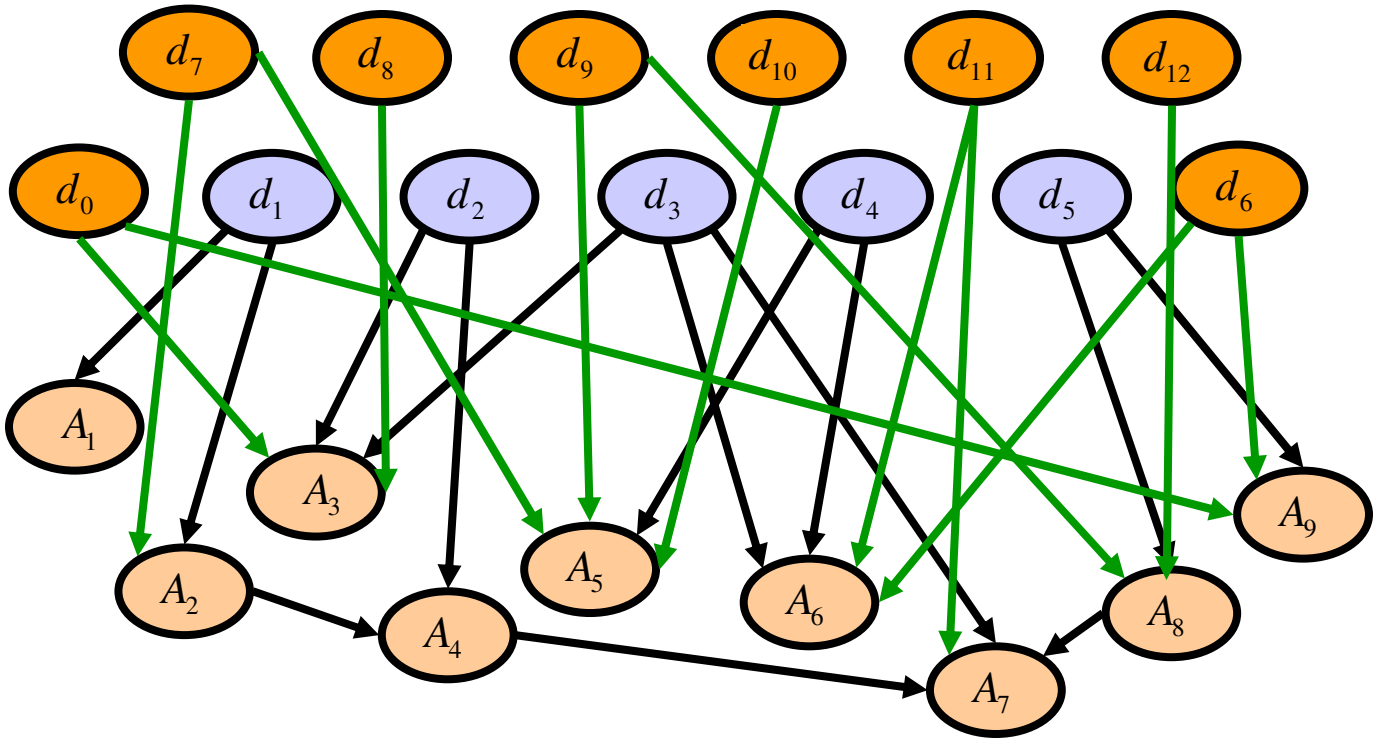
Constructing the CDG Engine



Problem: not feasible (and not necessary) to use all directives

Solution: perform feature selection to reduce their number

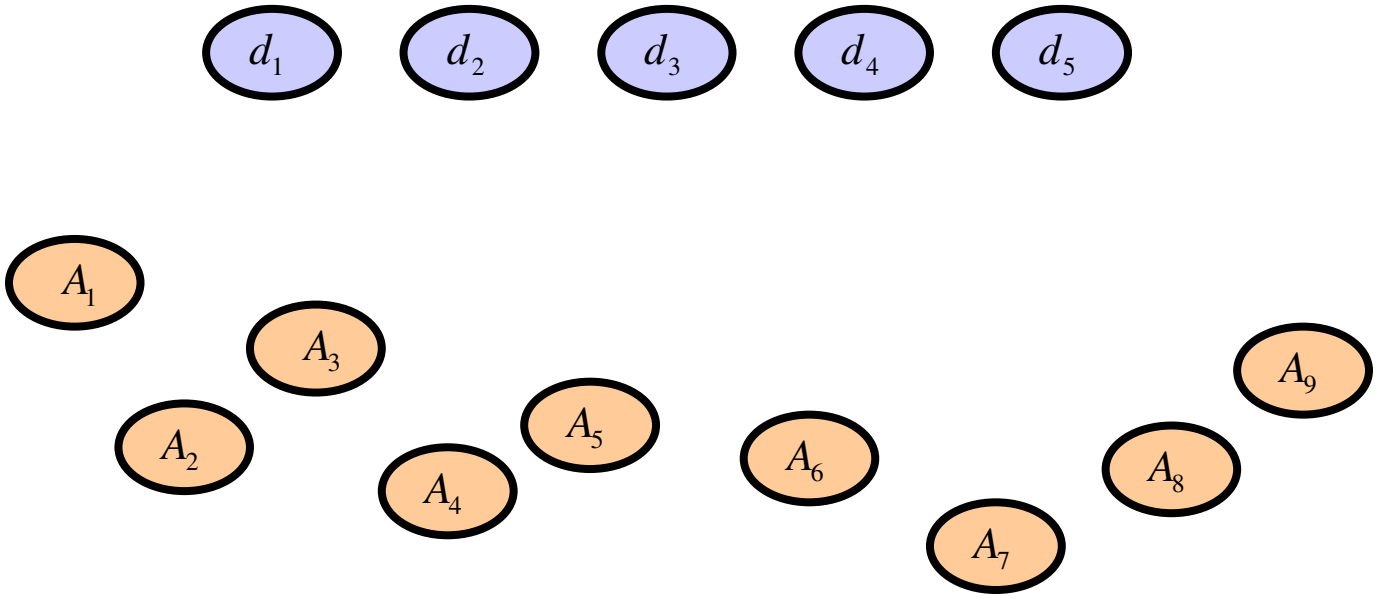
Constructing the CDG Engine



Problem: not feasible (and not necessary) to use all directives

Solution: perform feature selection to reduce their number

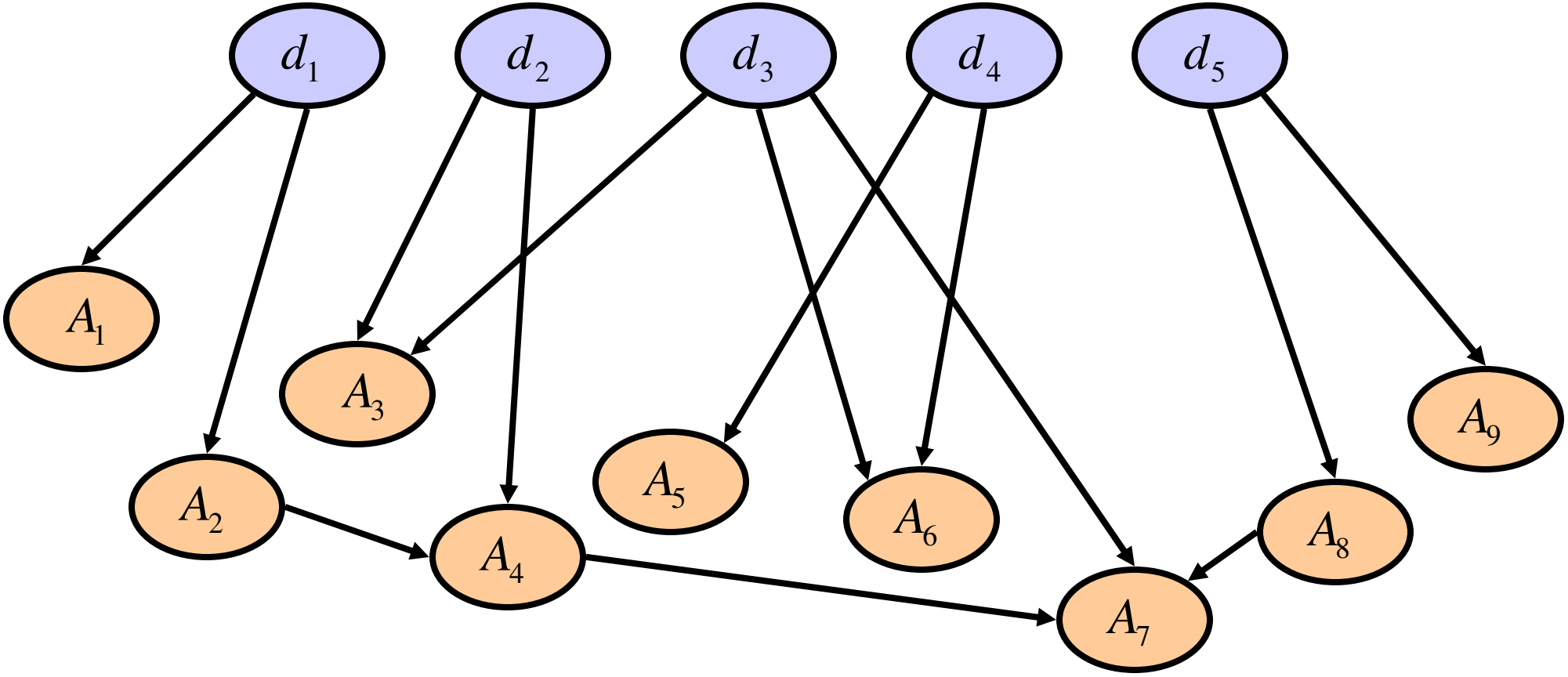
Constructing the CDG Engine



Problem: not feasible (and not necessary) to use all directives

Solution: perform feature selection to reduce their number

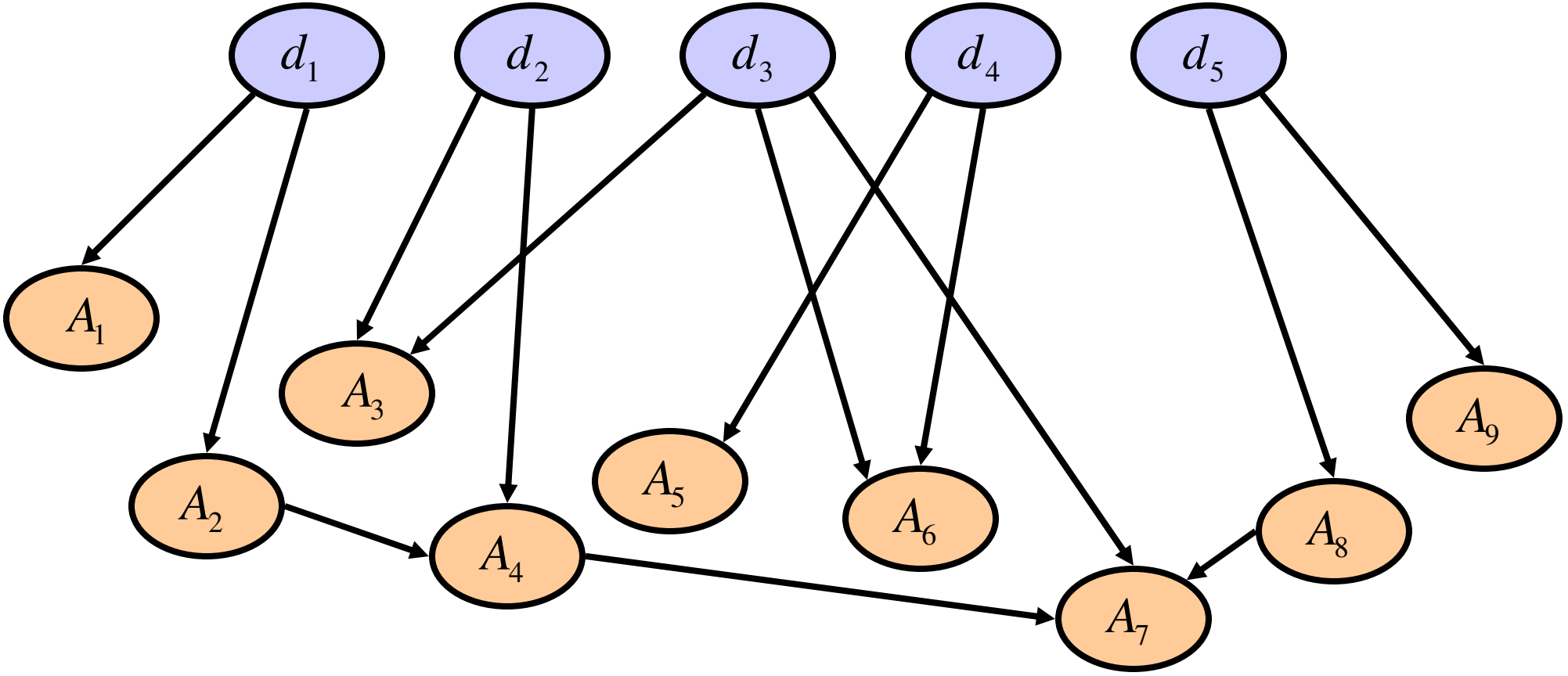
Constructing the CDG Engine



What is the network structure?

Goal - specify arcs between nodes

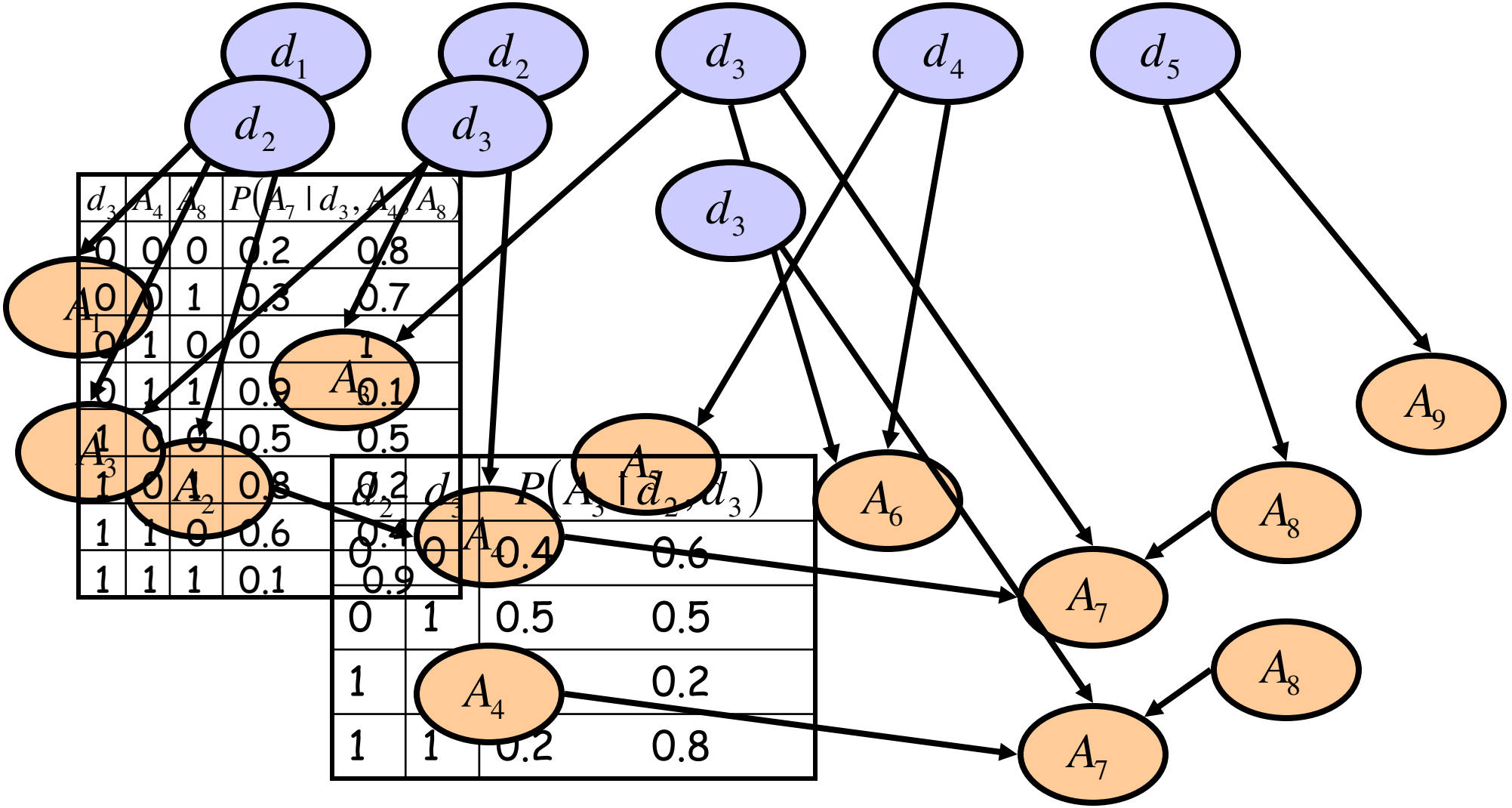
Constructing the CDG Engine



What are the probability distributions?

Goal - specify probabilities of nodes given their parents

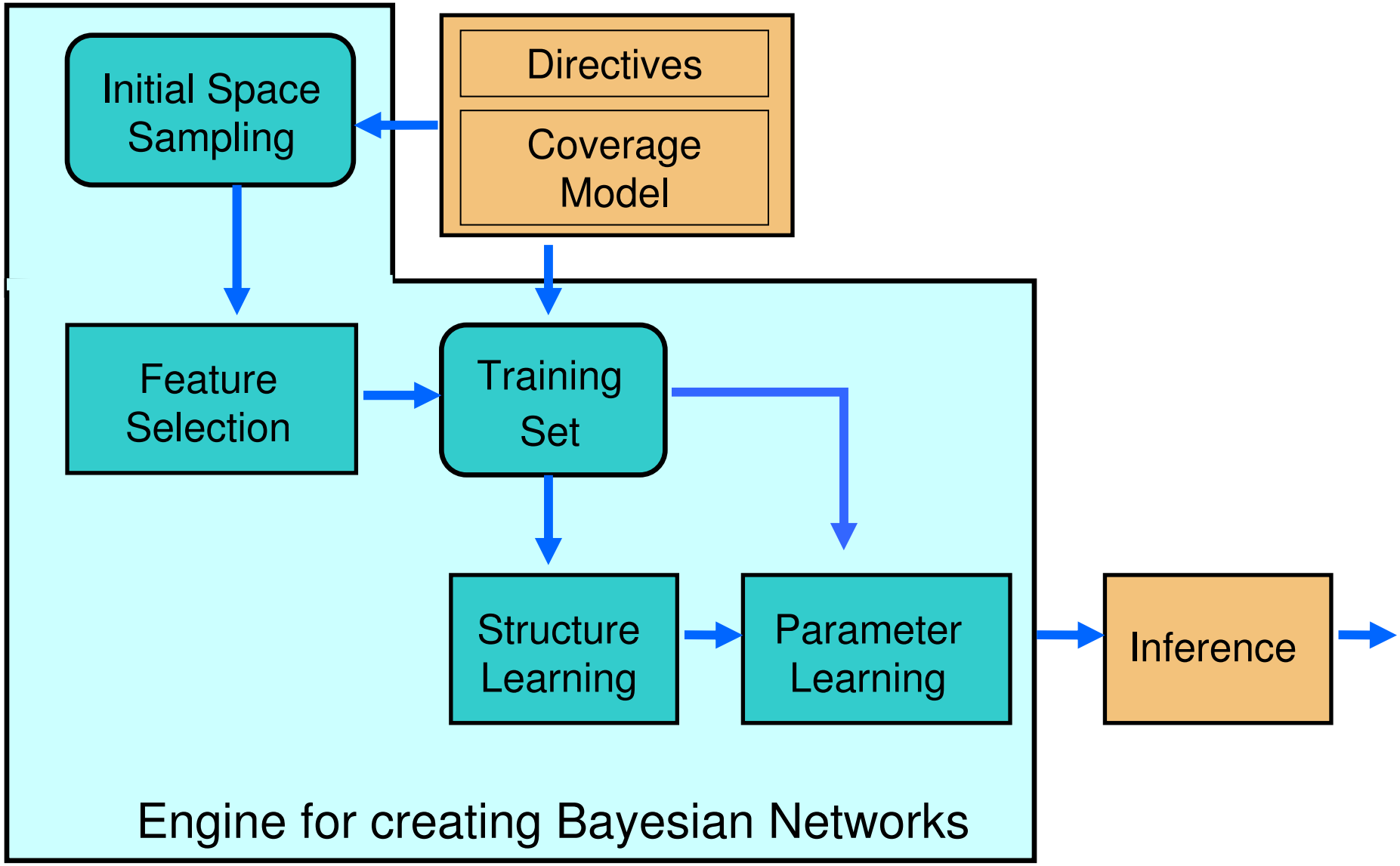
Constructing the CDG Engine



Booster Components

- Feature Selection
  - What directives to use?
    - Reduce number of directives
    - Choose "good" directives
- Structure Learning
  - How the Bayesian network looks like?
- Parameter Learning
  - What are the conditional probabilities?

Engine for Creating Bayesian Networks



Feature Selection

- Reminder – feature selection goals:
  - Reduce number of directives (dozens to few)
  - Use directives that have influence
- Difficulties in CDG:
  - Noise
  - Dominant values
- Score strength of dependency between pairs (directive, attribute)
  - Choose "high quality" subset
    - Tradeoff: affects many vs. high score on single
  - Use mutual information
    - Measure from information theory
    - Quantity that measures mutual dependency of two random variables

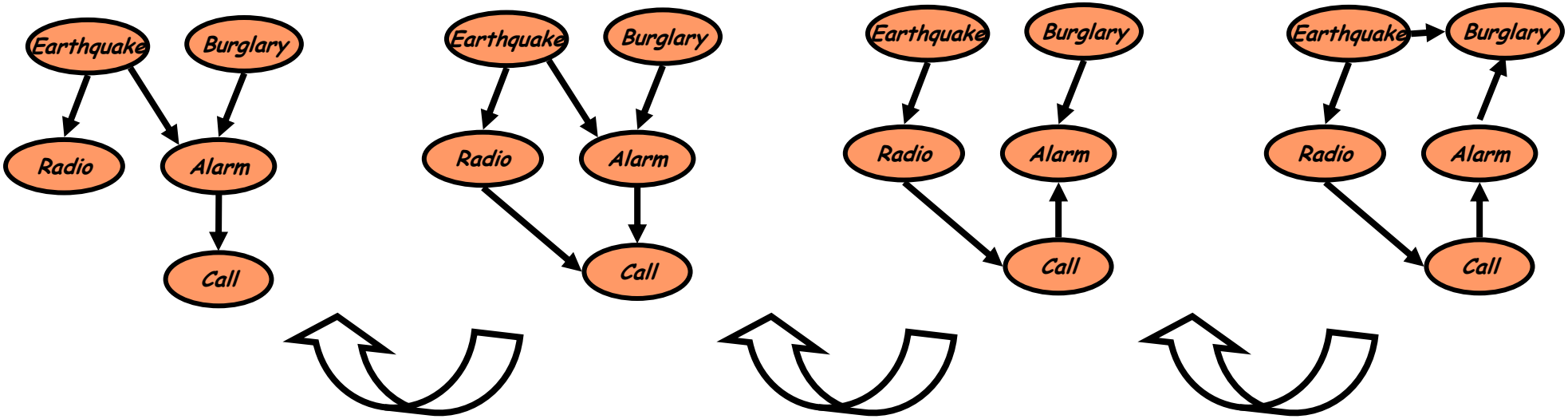
$$I(X;Y) = \sum_{x,y} P(x,y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right)$$

Structure Learning

- Difficult task
  - Results depends on scoring function
  - Slow convergence and no guarantee for optimality
  - Requires large amount of fully observed data
  - Generic algorithms vs. application based algorithms
- CDG is even harder…
  - Data is not fully observed (directives are weighted)
  - Data is noisy and there are dominant values
  - No specific algorithm that fits our setting
- CDG booster
  - Uses generic structure learning algorithms
    - Observed data – use directives $P(X = x_i) = 1, P(X \neq x_i) = 0$
  - Modifications to fit CDG setting and improve results

Structure Learning, cont.

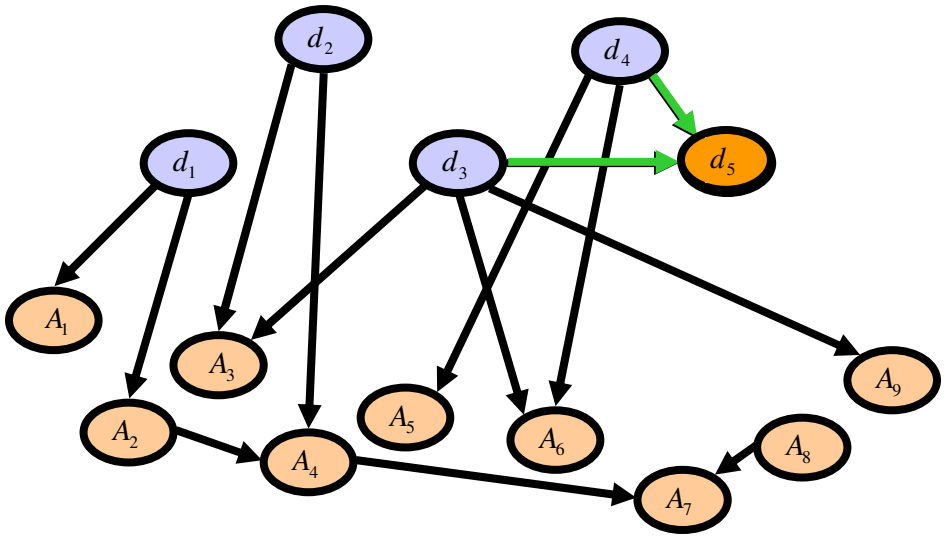
- Generic algorithms
  - Define score for a given network
  - Search in structure space to improve score
  - Use various types of greedy search till convergence



- Biggest problem - requires fully observed data!

Modifications and Improvements – Pruning

- Algorithms generated arcs between nodes representing directives
  - Artifacts of noise and algorithms behavior
  - Not natural in verification environment
- Solution: prune arcs between directives
  - Improve results (generalize better)
  - Performs additional feature selection (orphan nodes)



Modifications and Improvements – Quantization

- Overcome problem of using only fully observed data
  - Directives are given deterministic values, not probabilities
- Quantized directives – transformation to fully observed data (example):
  - Given a directive $X \in \{1,2,3\}$
  - Define Z with domain $\{z_1, z_2, ..., z_k\}$ where $z_i = P_i(X)$
    - Namely – each value is a distribution over the directives

$$z_i = \{P(X=1), P(X=2), P(X=3)\}$$

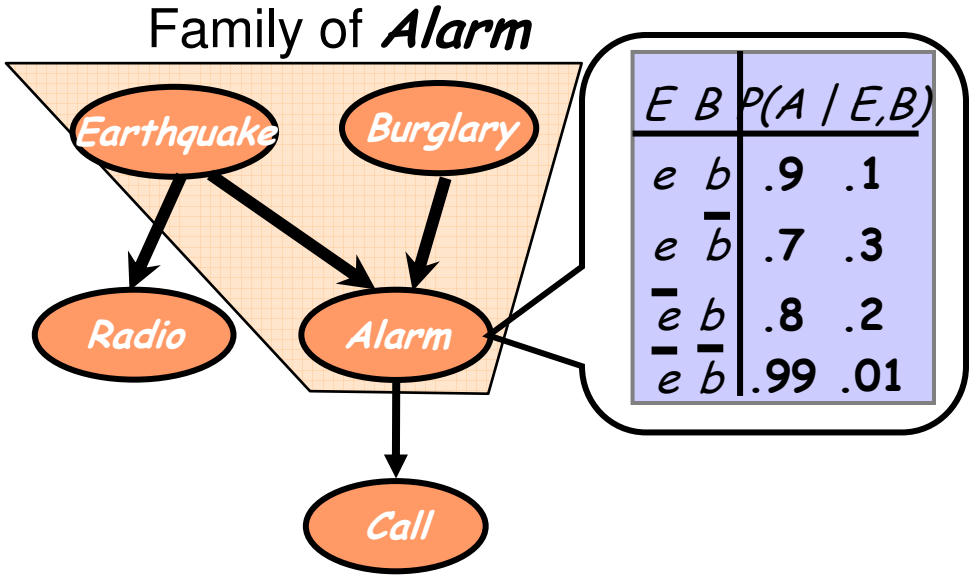
  - Transformation to uniform probabilities:

$$z_1 = \{1,0,0\}, z_2 = \{0,1,0\}, z_3 = \{0,0,1\}, z_4 = \left\{\frac{1}{2},\frac{1}{2},0\right\}, z_5 = \left\{\frac{1}{2},0,\frac{1}{2}\right\}, z_6 = \left\{0,\frac{1}{2},\frac{1}{2}\right\}, z_7 = \left\{\frac{1}{3},\frac{1}{3},\frac{1}{3}\right\}$$

- Learn structure with Z
- Use same structure with X instead of Z
- Training data is fully observed (Z is known)

Parameter Learning

- Standard estimation algorithm
  - EM (expectation – maximization)
  - Handles unobserved data (distribution of directives)
  - Training set is of the form
    - (Directives, event)
    - In the network below: $\{P(E), P(B), R, A, C\}$
  - Convergence to local minima
- Yet, works well



Outline

- Simulation Based Verification Environment
- The Coverage Booster
- CDG + Bayesian Networks
- Booster Structure
  - Feature Selection
  - Structure Learning
  - Parameter Learning
- Experiments
- Conclusions and Future Work

Experiments – Pipe Model (z10's IFU)

- Instruction Fetch Unit in IBM's latest system z (mainframe) processor
- Handles branch prediction tasks
  - Branch address
  - Branch results (taken/not taken)
  - Branch target
- Pipe model – snapshot of the processor pipeline
  - 13 attributes
    - Stares of pipeline stages
    - Flags relating branches and recycles
  - Coverage space – 139968 events
  - Legal space – 54000 events
  - Interesting space – 27162 events

Results

- Feature selection
  - Reducing 22 directives to five or less
  - Domain sizes two to five
- Parameter learning and inference
  - Using standard algorithms
- Main focus: learning the structure
  - Use existing algorithms (K2, gs, mcmc, structural EM)
  - Evaluate using regression as baseline
    - Regression – collection of directives manually designed by verification team
    - Show boosting
    - Avoid attempt to hit easy-to-reach events

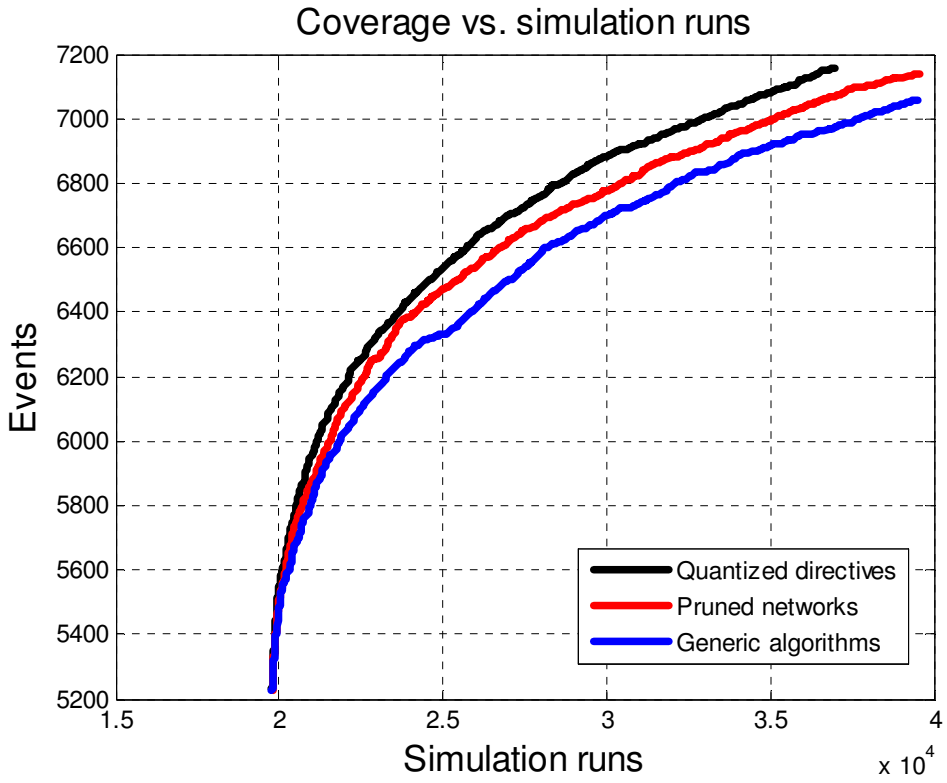
Experiments Description

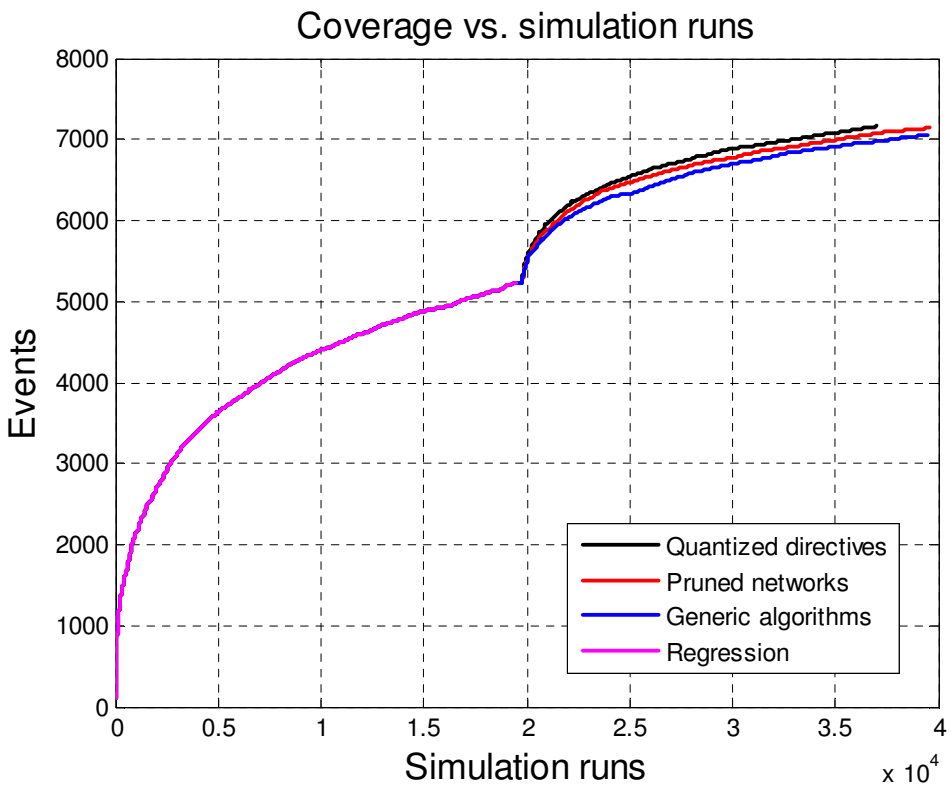
- Experiment 1 - Use algorithms as is
  - Use delta probabilities, i.e. directives have distributions of the form

$$P(X = x_i) = 1, P(X \neq x_i) = 0$$

  - Five directives
  - Compare to naïve network – connect nodes that have high mutual information
- Experiment 2
  - Prune networks - remove arcs between directives
- Experiment 3
  - Use quantized directives
  - Prune arcs between directives (as in Experiment 2)
- In all experiments use regression as baseline

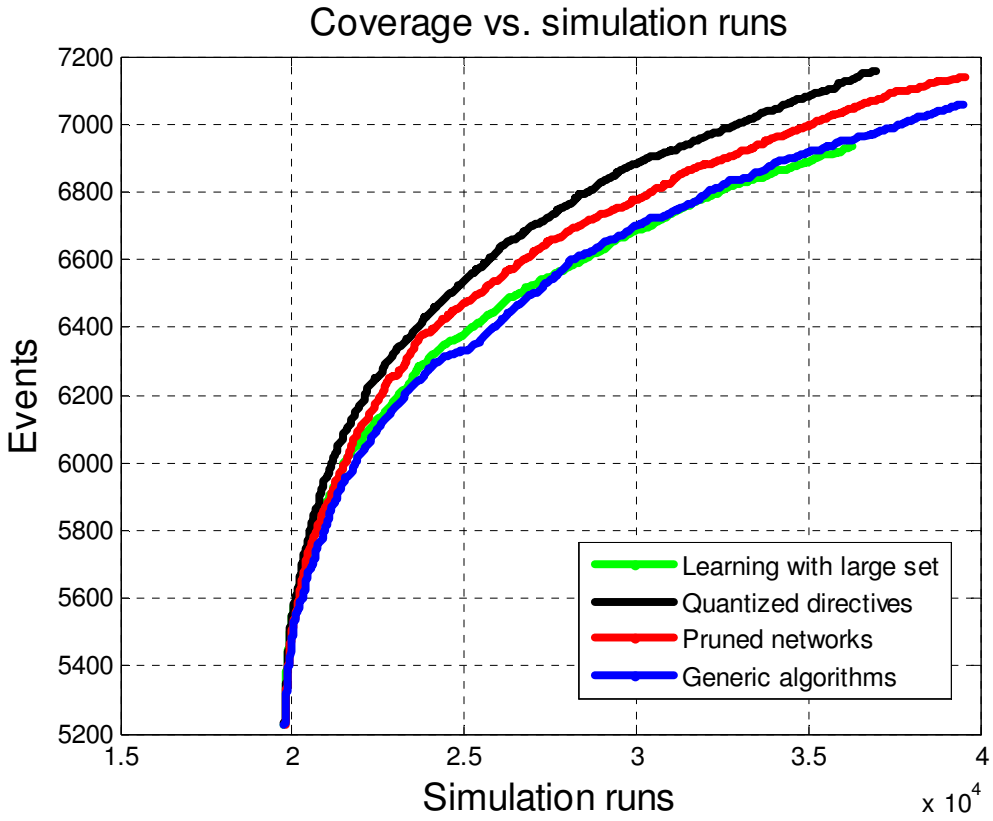
Structure Learning Results





Structure Learning with Large Set

- Use large set of directives to test performance in presence of noise
- 11 Directives instead of five
- Pruning performs as feature selection
- Results are slightly worse
  - More noise is inserted to the system
  - Algorithms perform worse when learning large networks
- Solution: relearn with remaining directives



More (Interesting) Results

- Various algorithms performed differently
  - Different directives orphaned
    - In Experiment 2 (pruned networks), three or four directives remain
    - In Experiment 4 (learning with a large set), three to five directives remain
  - Different ability to generate directives
    - Inability to perform inference
  - Different performance in tests
    - No "winner" algorithm

Summary

- An automatic method for boosting coverage was presented
- CDG engine construction require minimal domain knowledge
- Several heuristics and modifications were proposed to improve performance
  - Pruned networks
  - Quantized directives
  - Structure learning as means for feature selection
- Results on a real problem are encouraging

Future Work

- Extend/modify existing algorithms to better fit CDG setting
- Develop means to estimate quality of chosen subsets (directives)
- Develop methodology of sampling directives for the training set

The End