Probabilistic Feature Models

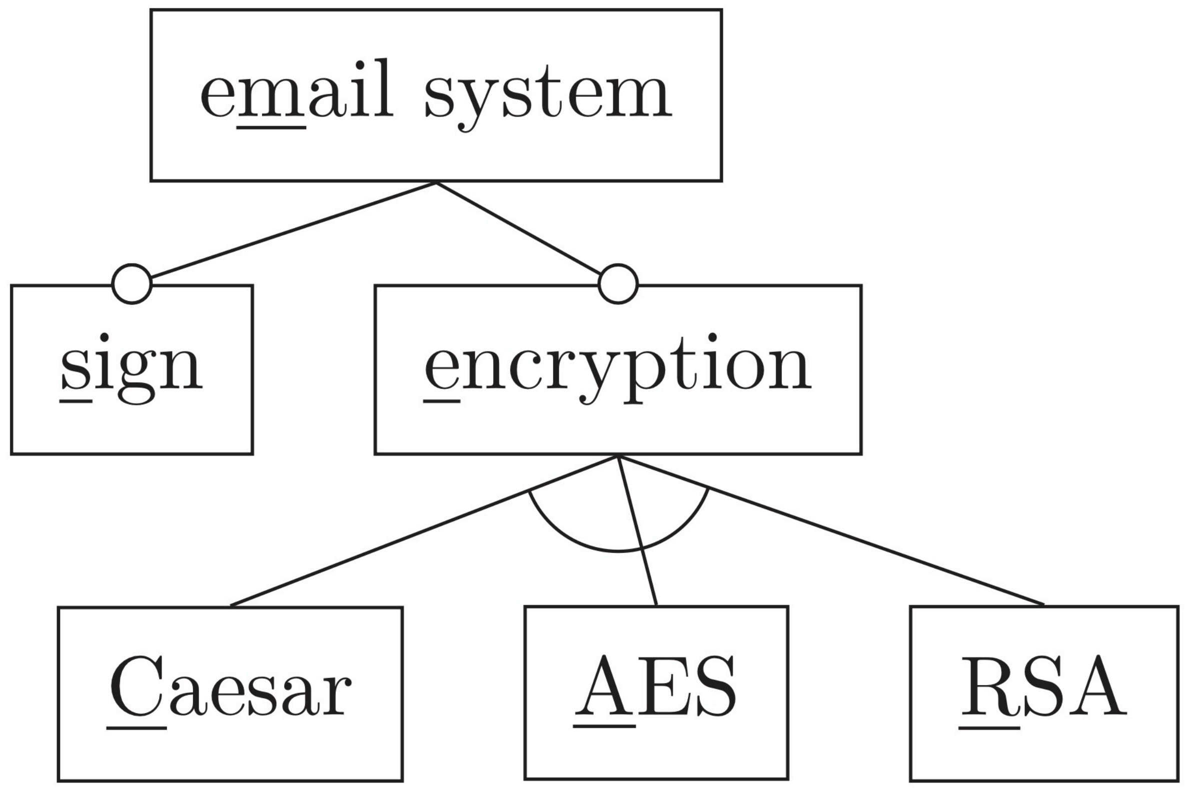
Nikolai Käfer, TU Dresden

FOSD Meeting – April 9, 2024

CPEC
CENTER FOR PERSPICUOUS COMPUTING
TECHNISCHE UNIVERSITÄT DRESDEN

How to represent real-world distributions over feature configurations?

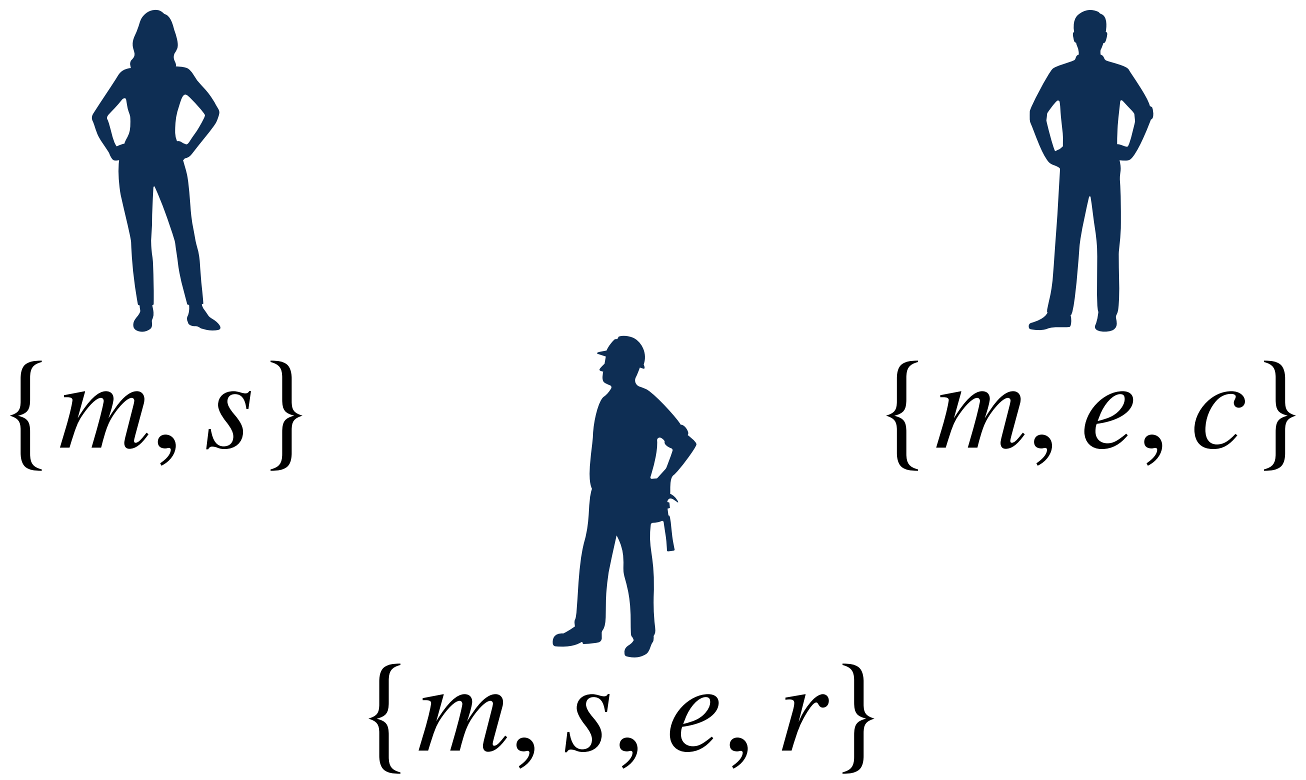
feature model of a software system



$s \rightarrow \neg c$

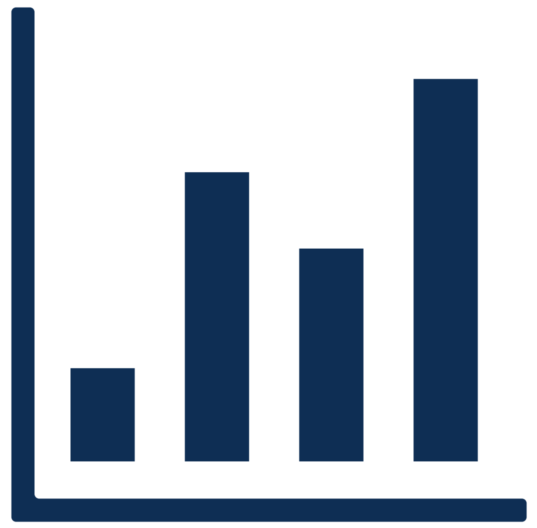
defines **valid configurations**

usage of the system "in the wild"



distribution over valid configurations

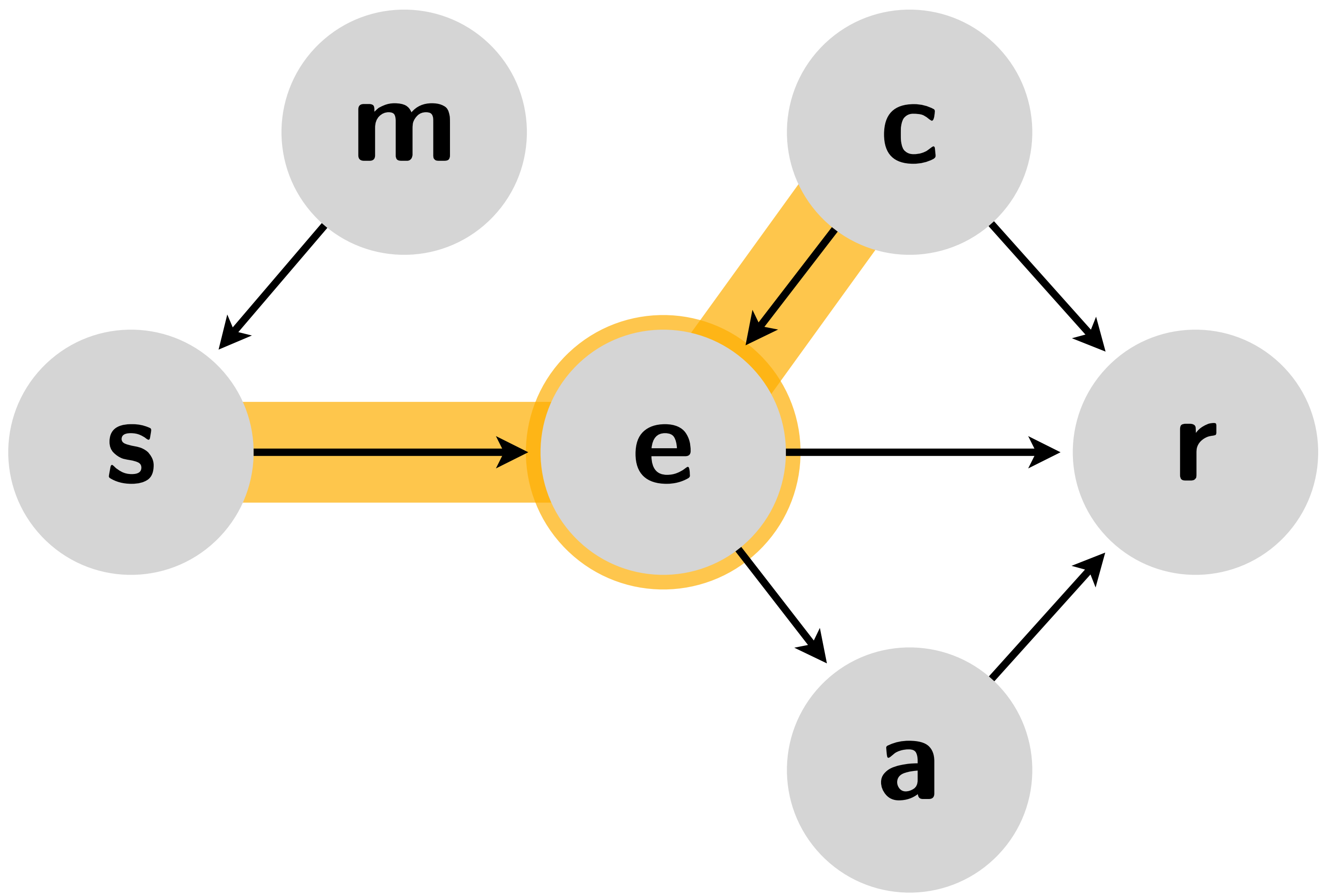
configuration **samples** / statistics



observable **data**

Representation as Bayesian Network (BN)

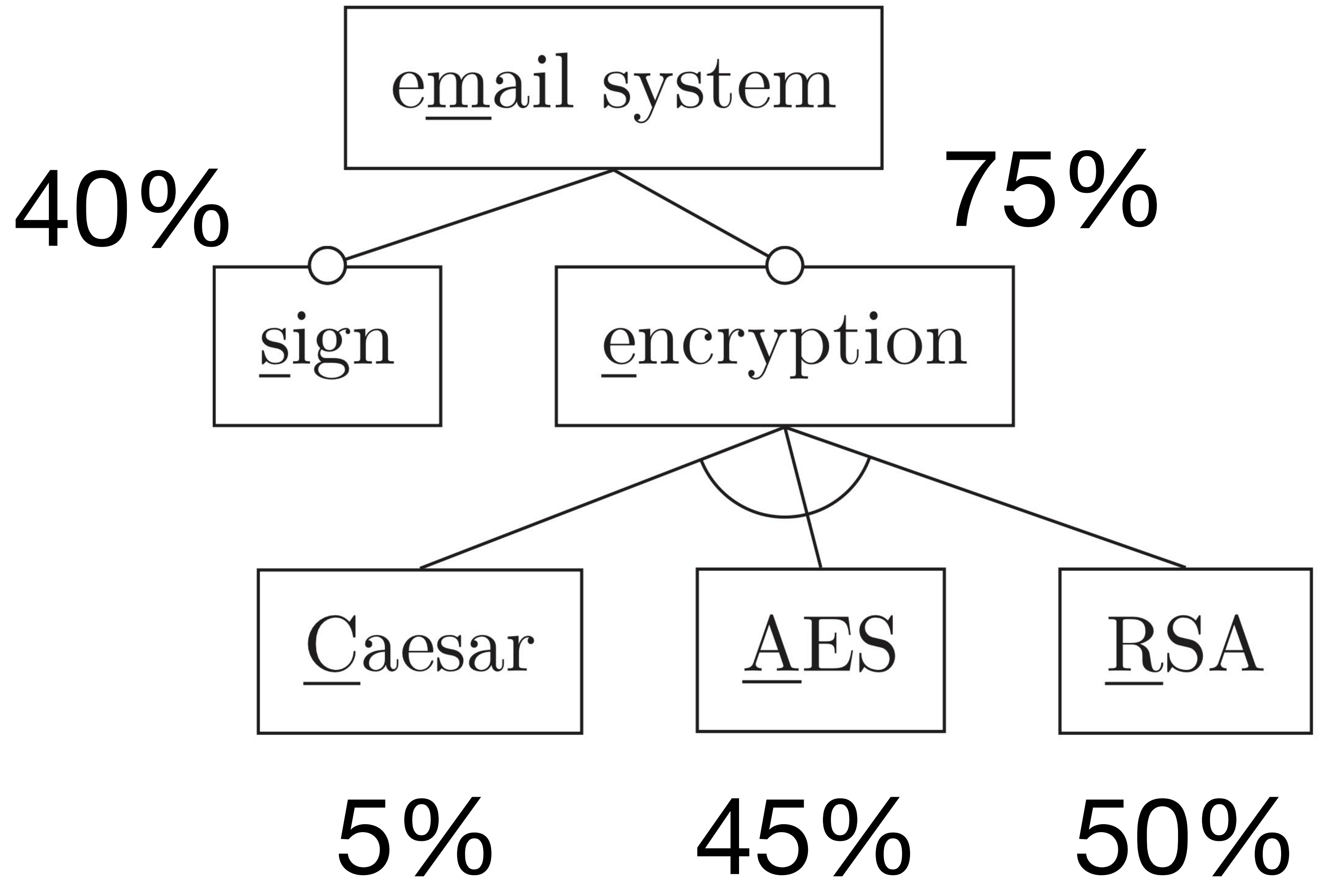
- **factorised representation** of a probability distribution
- local **conditional probability distributions** for each node
- encodes **independencies** (more on this later)
- can be learned from data in two-steps: **structure learning** and **parameter learning**
- structure is **not unique**
- **invalid configurations** could be learned

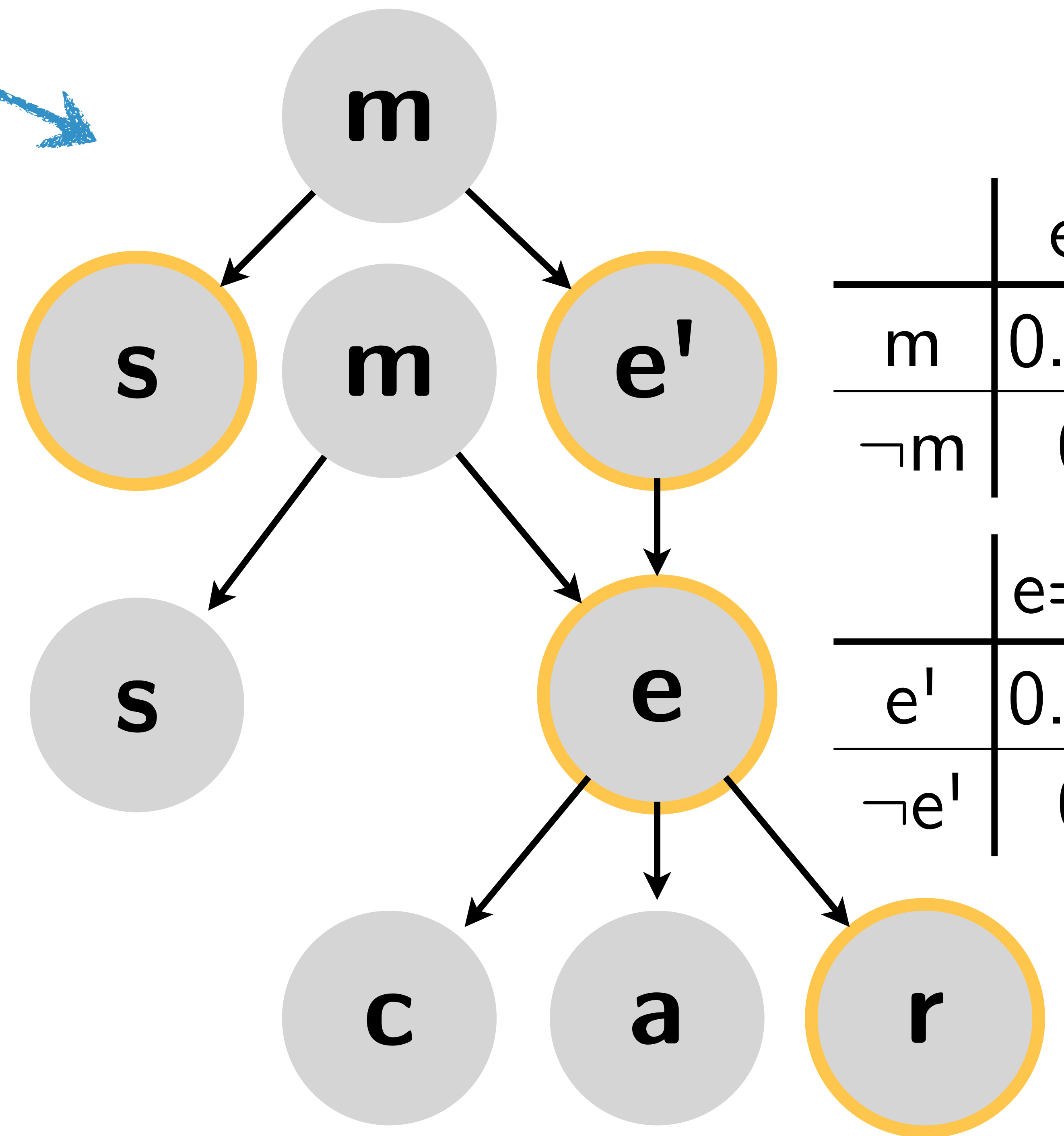


		e	¬e
s	c	1	0
s	¬c	0.7	0.3
¬s	c	0	1
¬s	¬c	0.2	0.8

$$\Pr(e \mid s, \neg c) = 0.7$$

Probabilistic Feature Models





	s	¬s
m	0.4	0.6
¬m	0	1

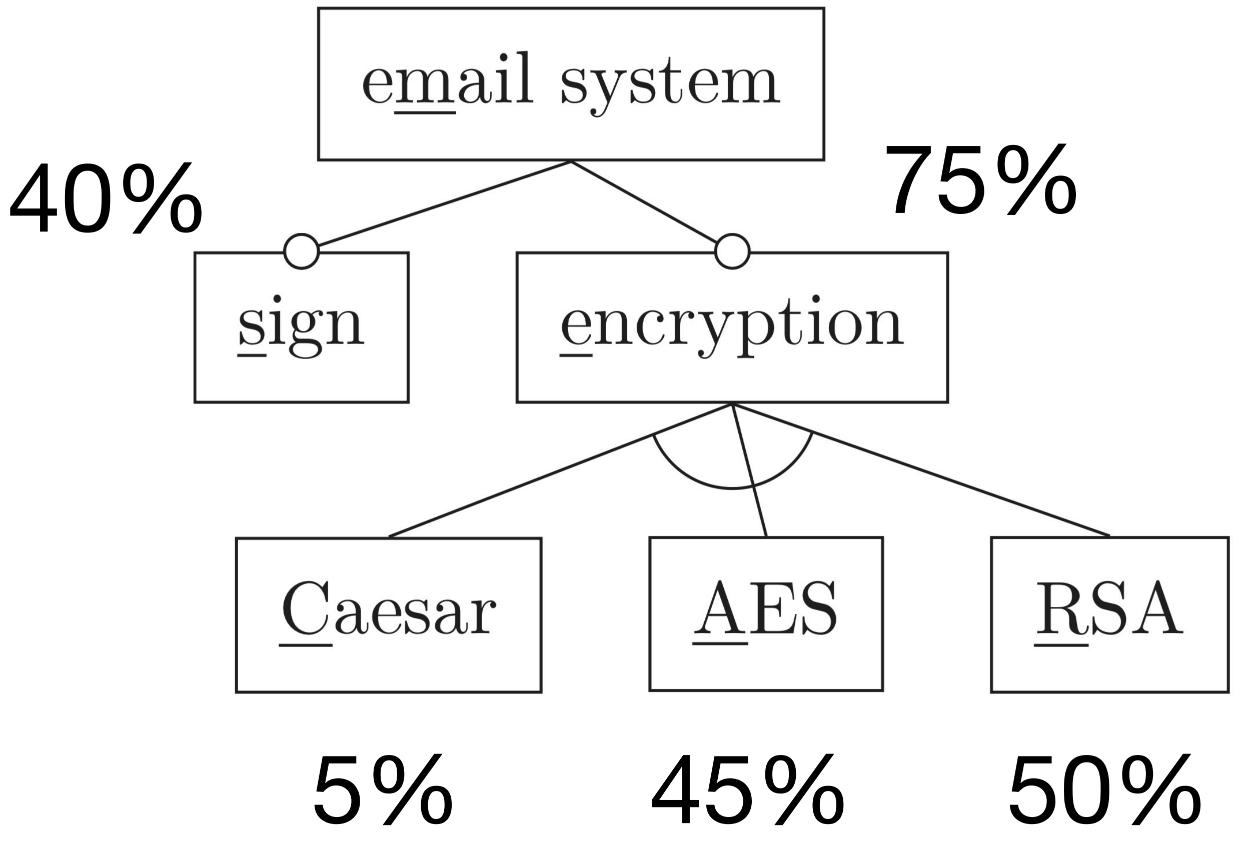
	e'	¬e'
m	0.75	0.25
¬m	0	1

	e=c	e=a	e=r	¬e
e'	0.05	0.45	0.5	0
¬e'	0	0	0	1

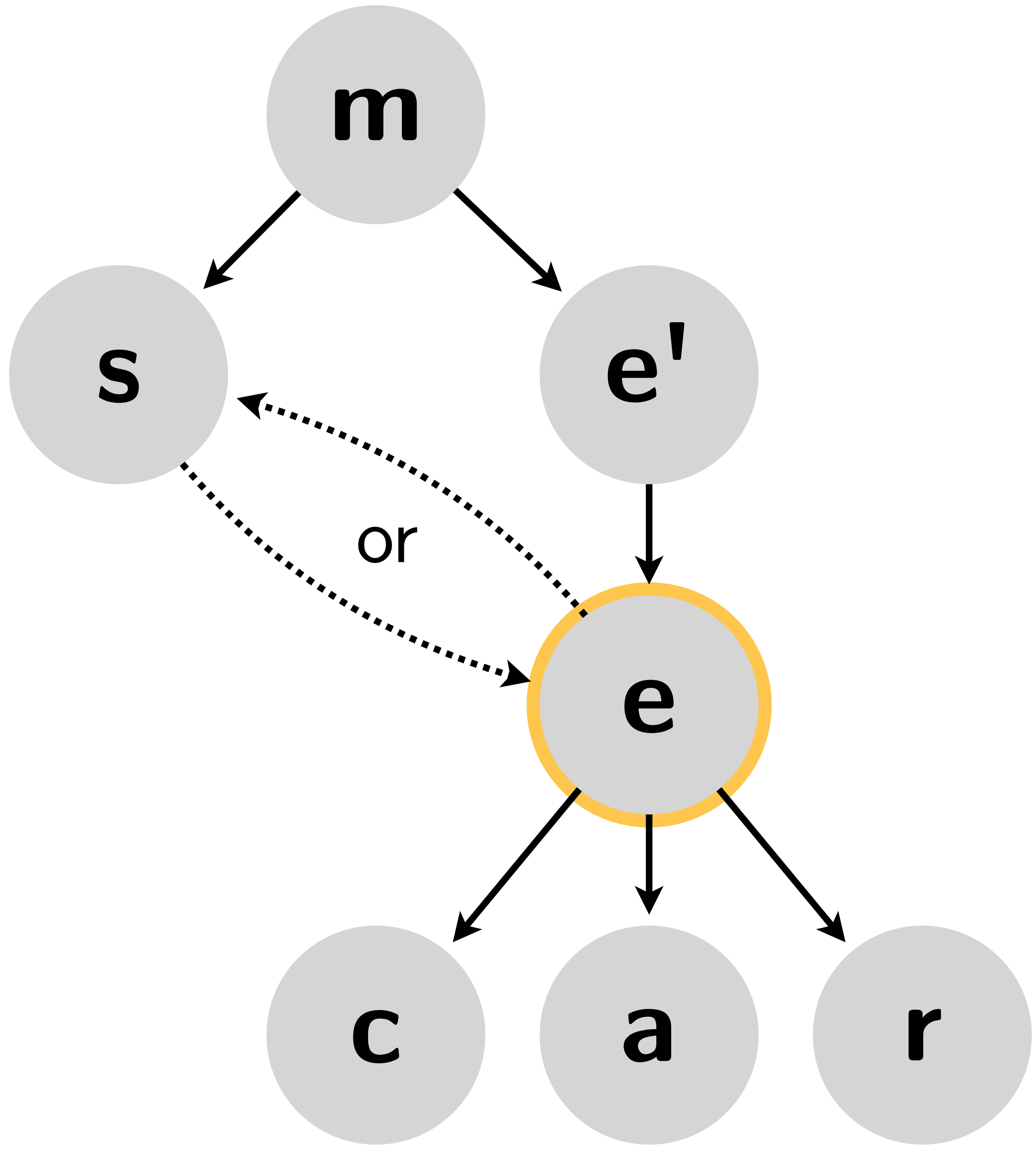
	r	¬r
e=c	0	1
e=a	0	1
e=r	1	0
¬e	0	1

- **Properties:**
  - Invalid configurations have probability zero
  - Valid configs have positive probability (if annotations are positive)

Dealing with side constraints



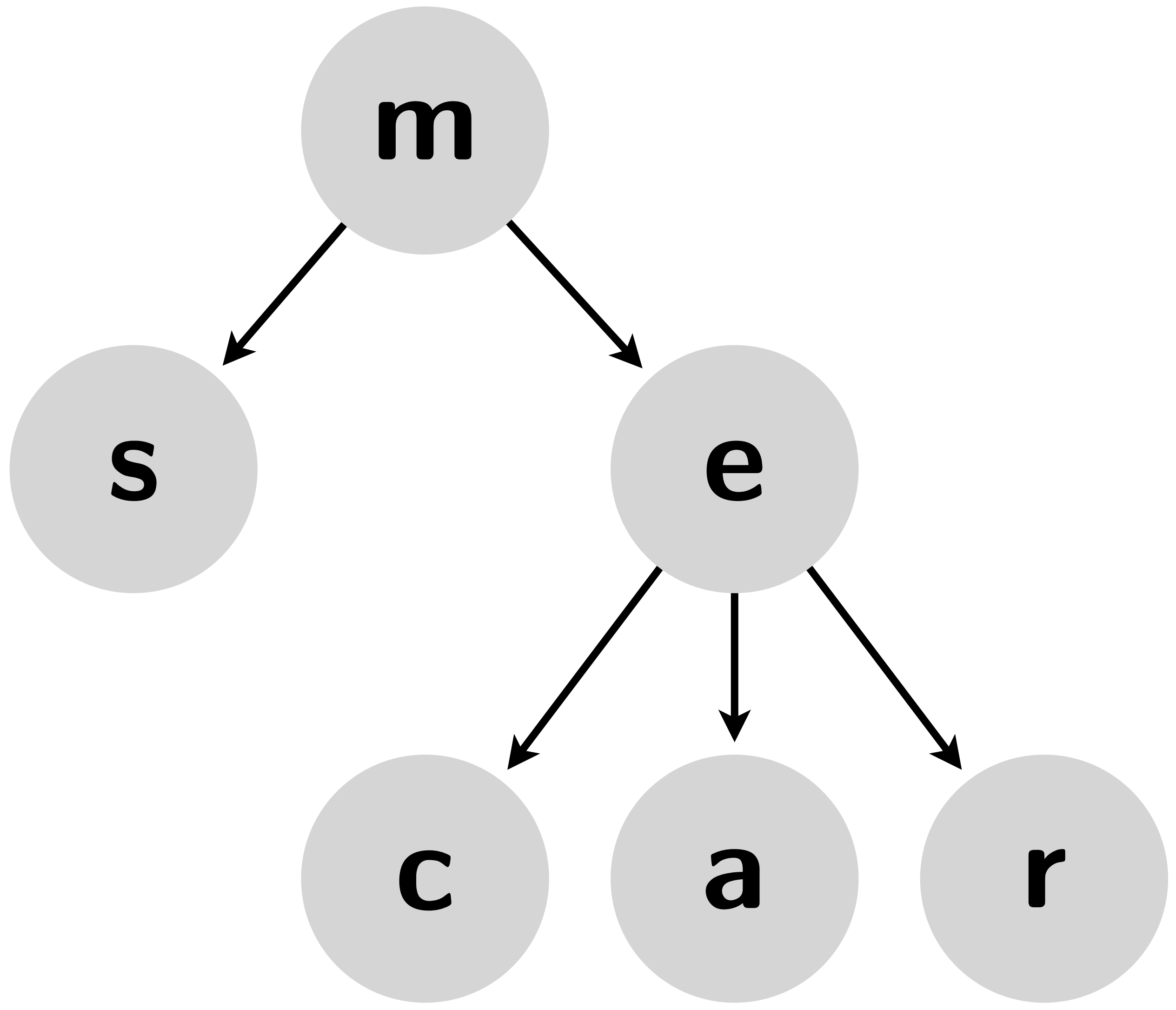
		e=c	e=a	e=r	¬e
e'	s	0	0.47	0.53	0
¬e'	s	0	0	0	1
e'	¬s	0.05	0.45	0.5	0
¬e'	¬s	0	0	0	1



$$s \to \neg c$$

Independencies

- BN structure determines some (conditional) independencies
- **($s \perp e \mid m$):** features **s** and **e** are conditionally **independent** given **m**
- **Problem:** real-world distributions may **not** exhibit all of these independencies
- "no dependency" in feature model ≠ "independency" in distribution



Summary: Probabilistic Feature Diagrams

- Feature diagrams with **probabilistic annotations**
- Formal **BN semantics** with some guarantees
- Allowing **arbitrary side constraints** provides (interesting?) challenges
- Probably **not** the answer to the opening question:

How to represent real-world distributions over feature configurations?