

Artificial Intelligence

Knowledge, Reasoning and Knowledge Representation; Uncertain Knowledge and Reasoning

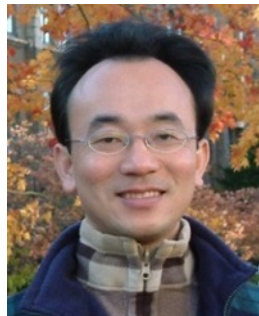
1131AI03

MBA, IM, NTPU (M5276) (Fall 2024)

Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/paj-zhhi-mya>



Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week Date Subject/Topics

1 2024/09/10 Introduction to Artificial Intelligence

2 2024/09/17 Mid-Autumn Festival (Day off)

3 2024/09/24 Artificial Intelligence and Intelligent Agents; Problem Solving

**4 2024/10/01 Knowledge, Reasoning and Knowledge Representation;
Uncertain Knowledge and Reasoning**

5 2024/10/08 Case Study on Artificial Intelligence I

6 2024/10/15 Machine Learning: Supervised and Unsupervised Learning

Syllabus

Week Date Subject/Topics

7 2024/10/22 The Theory of Learning and Ensemble Learning

8 2024/10/29 Midterm Project Report

9 2024/11/05 Self-Learning

10 2024/11/12 Deep Learning, Reinforcement Learning

11 2024/11/19 Case Study on Artificial Intelligence II

12 2024/11/26 Deep Learning for Natural Language Processing

Syllabus

Week Date Subject/Topics

13 2024/12/03 Computer Vision and Robotics

**14 2024/12/10 Generative AI,
Philosophy and Ethics of AI and the Future of AI**

15 2024/12/17 Final Project Report I

16 2024/12/24 Final Project Report II

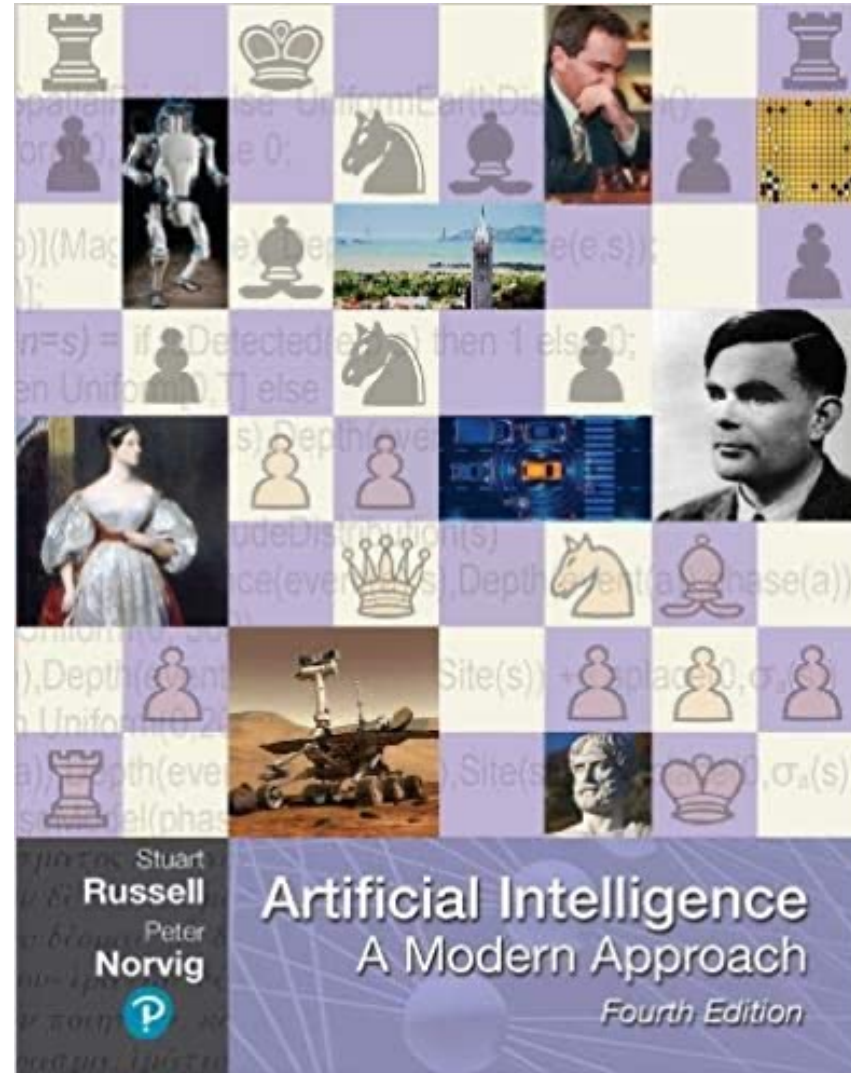
Knowledge, Reasoning and Knowledge Representation

Uncertain Knowledge and Reasoning

Outline

- **Knowledge and Reasoning**
 - **Logical Agents**
 - **First-Order Logic**
 - **Inference in First-Order Logic**
 - **Knowledge Representation**
 - **Knowledge Graph (KG)**
- **Uncertain Knowledge and Reasoning**
 - **Quantifying Uncertainty**
 - **Probabilistic Reasoning**
 - **Making Complex Decisions**

Stuart Russell and Peter Norvig (2020),
Artificial Intelligence: A Modern Approach,
4th Edition, Pearson



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

<https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997/>

Artificial Intelligence: A Modern Approach

1. Artificial Intelligence
2. Problem Solving
3. Knowledge and Reasoning
4. Uncertain Knowledge and Reasoning
5. Machine Learning
6. Communicating, Perceiving, and Acting
7. Philosophy and Ethics of AI

Artificial Intelligence: Knowledge and Reasoning

Artificial Intelligence:

3. Knowledge and Reasoning

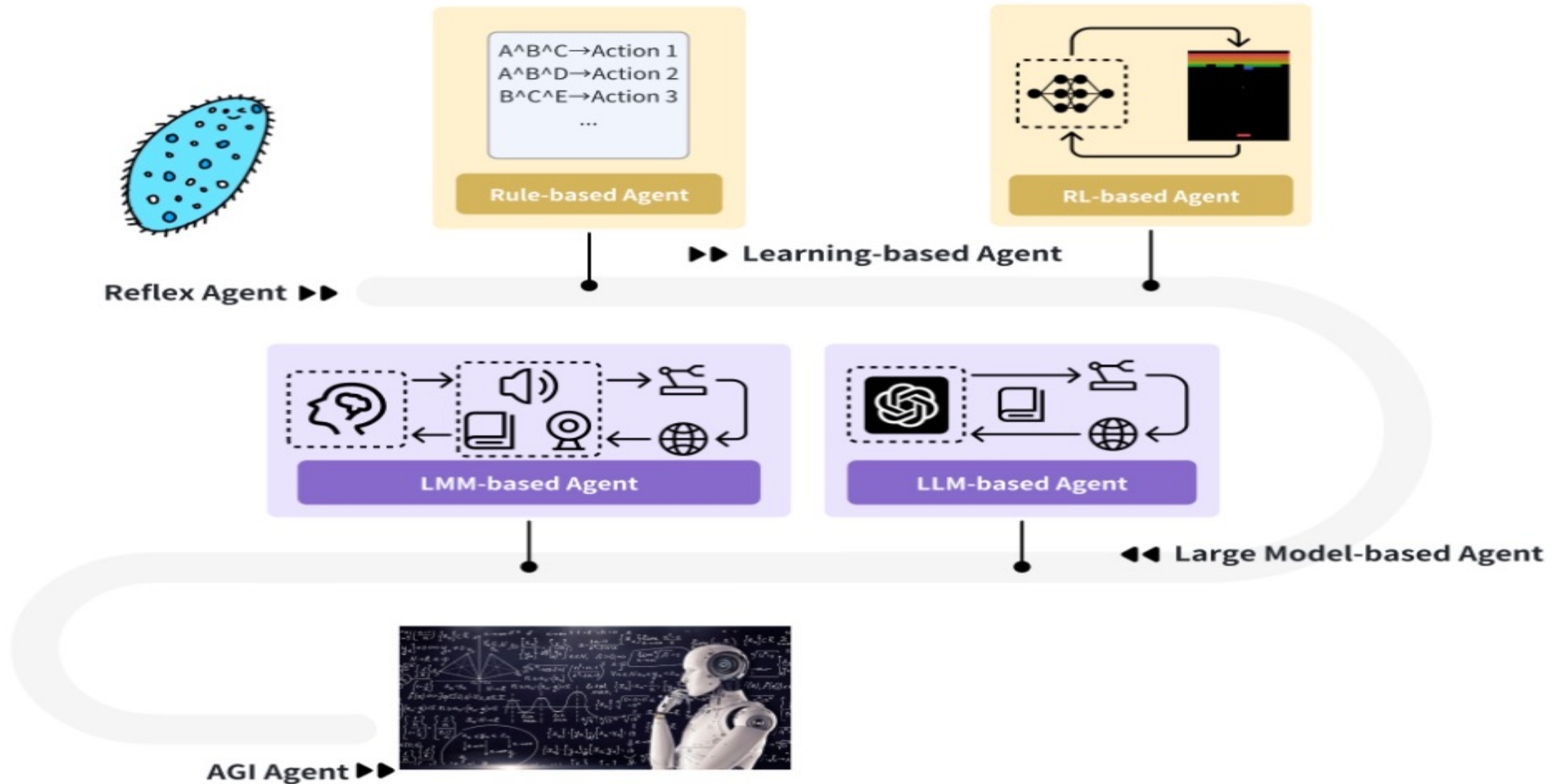
- **Logical Agents**
- **First-Order Logic**
- **Inference in First-Order Logic**
- **Knowledge Representation**
- **Automated Planning**

Intelligent Agents

4 Approaches of AI

<p>2. Thinking Humanly: The Cognitive Modeling Approach</p>	<p>3. Thinking Rationally: The “Laws of Thought” Approach</p>
<p>1. Acting Humanly: The Turing Test Approach <small>(1950)</small></p>	<p>4. Acting Rationally: The Rational Agent Approach</p>

Intelligent Agents Roadmap



AI Agents

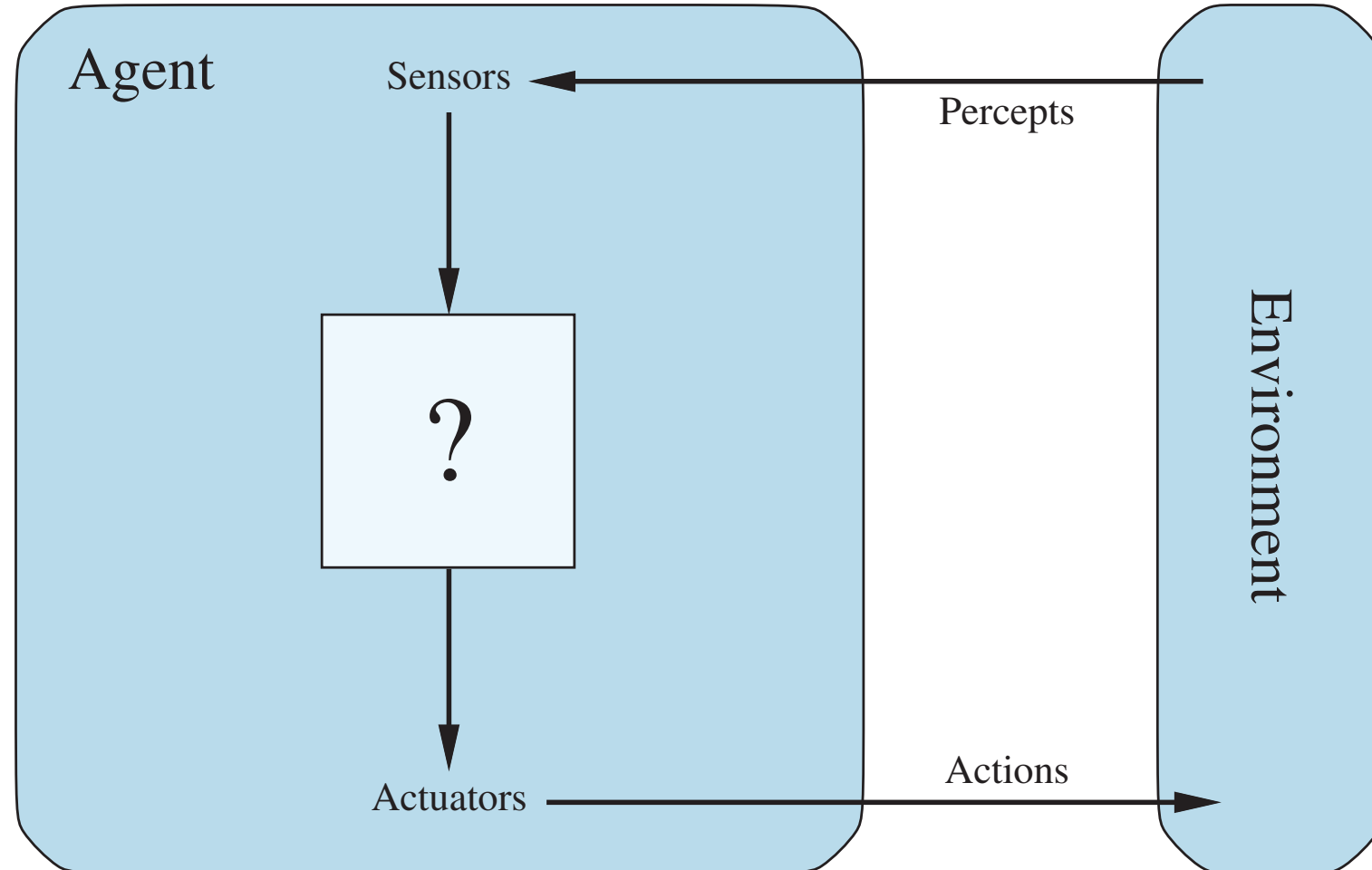
- **Traditional AI Agents**

- **Simple reflex agents**
- **Model-based reflex agents**
- **Goal-based agents**
- **Utility-based agents**
- **Learning agents**

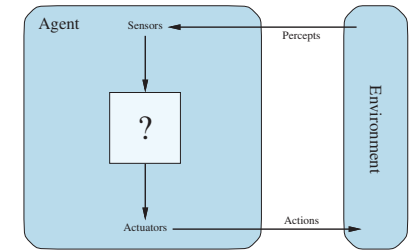
- **Evolution of AI Agents**

- **LLM-based Agents**
- **Multi-modal agents**
- **Embodied AI agents in virtual environments**
- **Collaborative AI agents**

Agents interact with environments through sensors and actuators



AI Agents



- **Definition:** An **AI agent** is an **entity** that **perceives** its **environment** and takes **actions** to achieve **goals**
- **Components:**
 - 1. Sensors:** Perceive the environment
 - 2. Actuators:** Act upon the environment
 - 3. Decision-making** mechanism: Process inputs and decide on actions

Reinforcement Learning (DL)

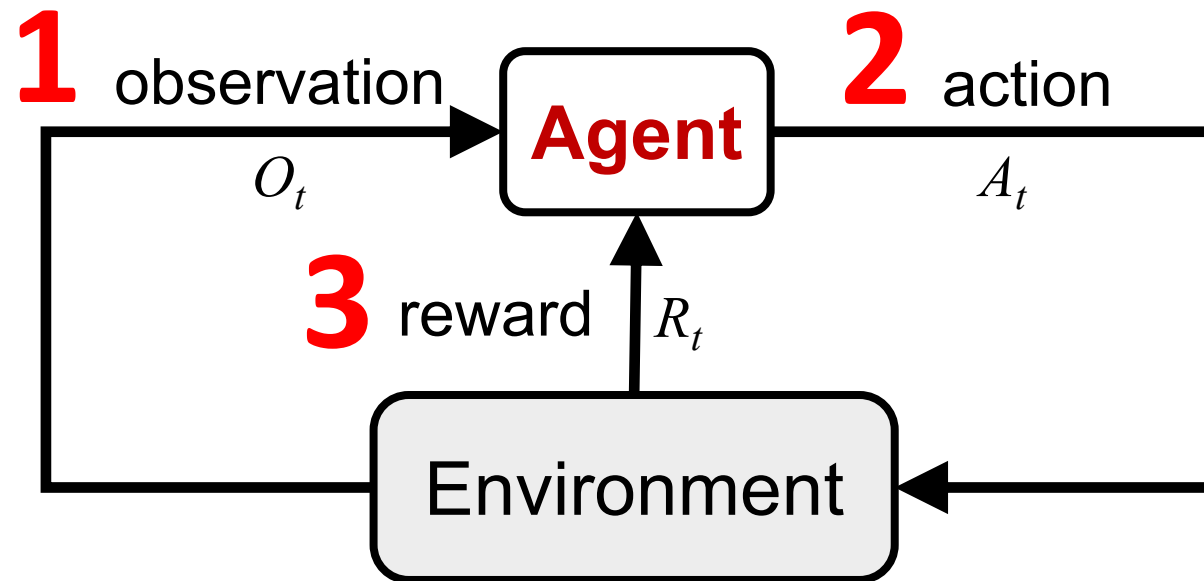
Agent

Environment

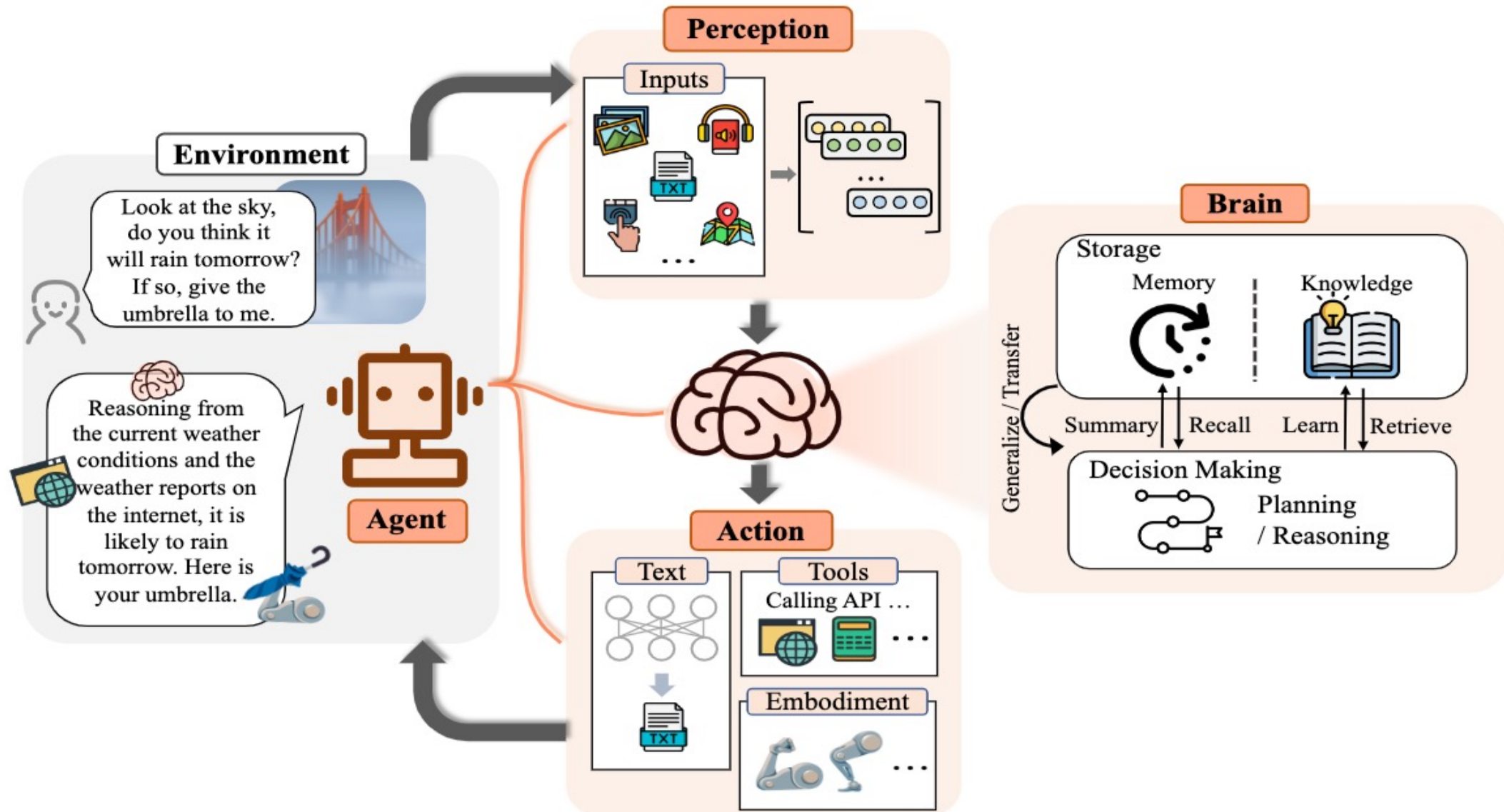
Reinforcement Learning (DL)



Reinforcement Learning (DL)



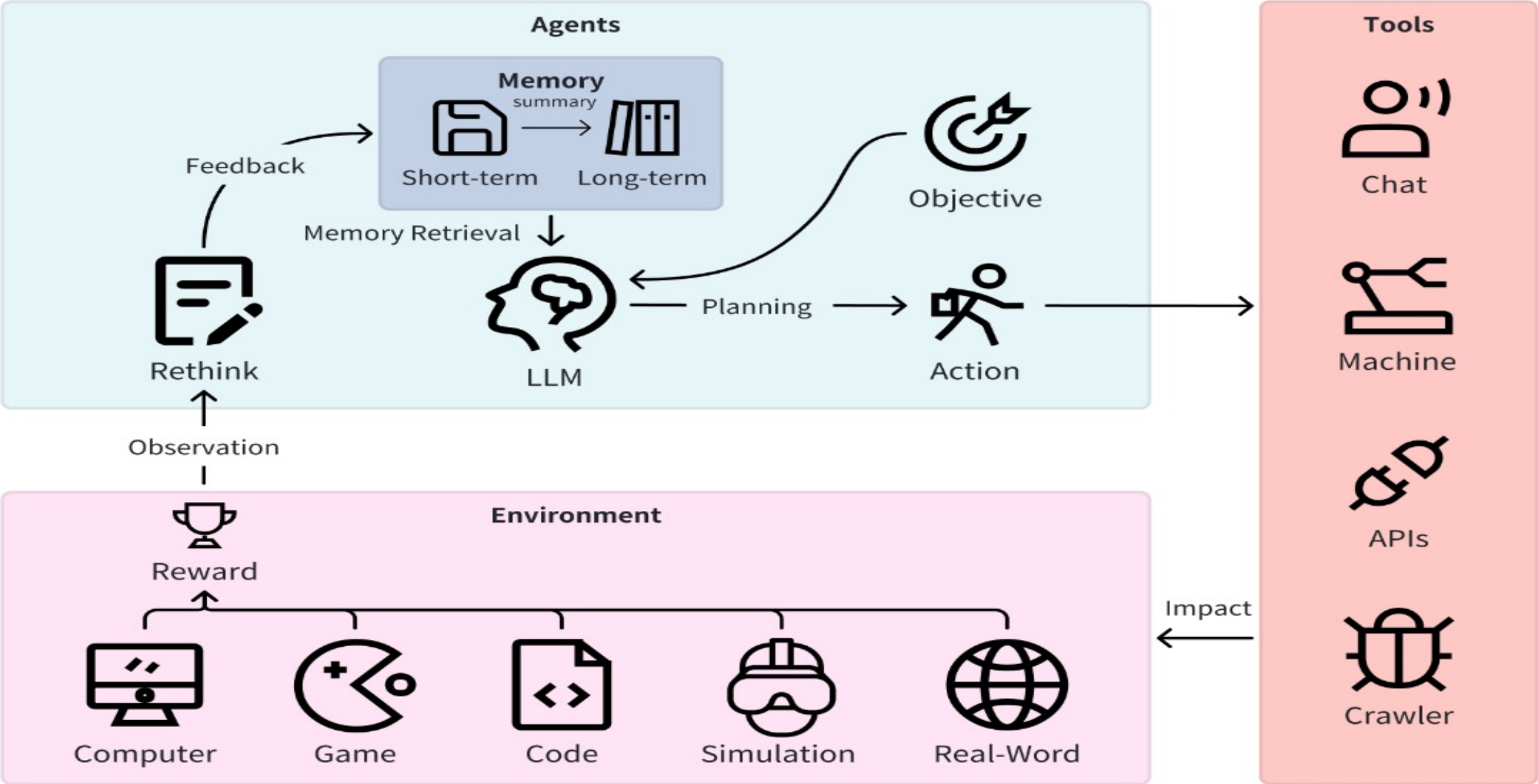
Large Language Model (LLM) based Agents



LLM-based Agents

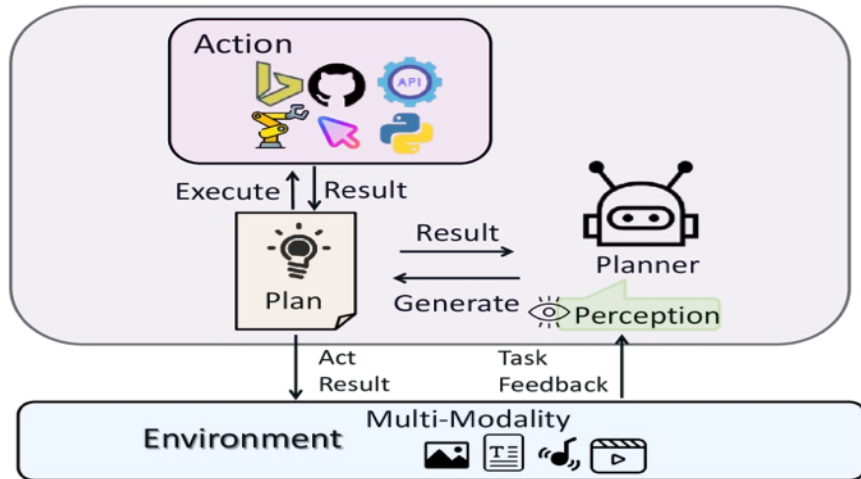
- **Definition:** **AI agents** that use **Large Language Models** as their **core decision-making** mechanism
- **Key Features:**
 - Natural language interface
 - Vast knowledge base
 - Ability to understand context and nuance
 - Generalize to new tasks with minimal additional training

LLM-based Agents

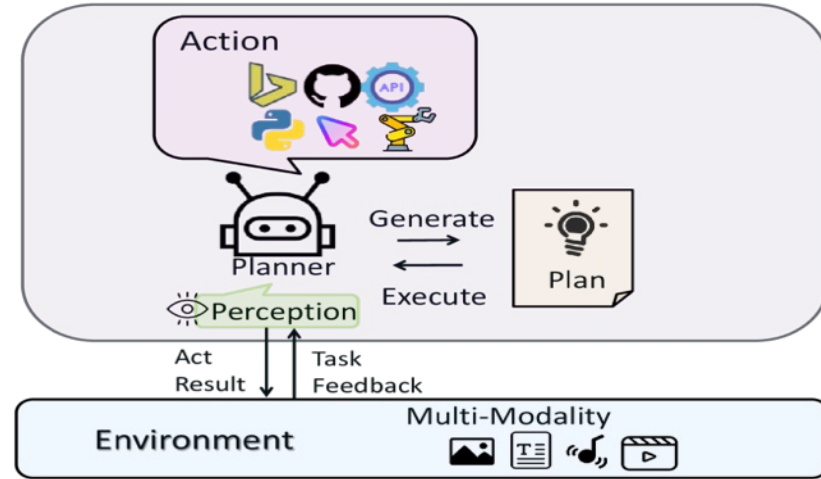


Source: Cheng, Yuheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang et al. "Exploring large language model based intelligent agents: Definitions, methods, and prospects." arXiv preprint arXiv:2401.03428 (2024).

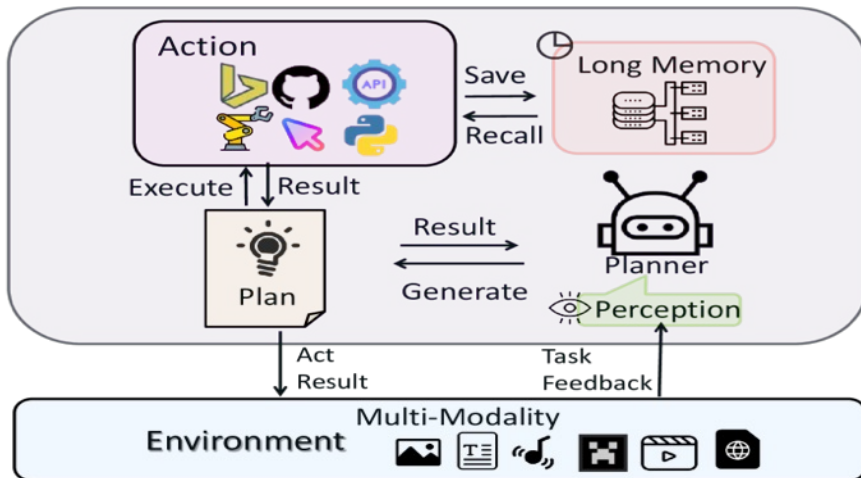
Large Multimodal Agents (LMA)



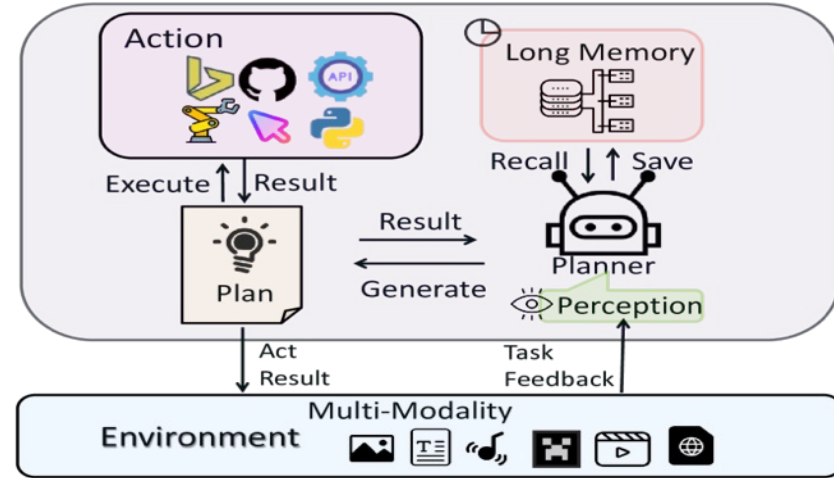
(a)



(b)

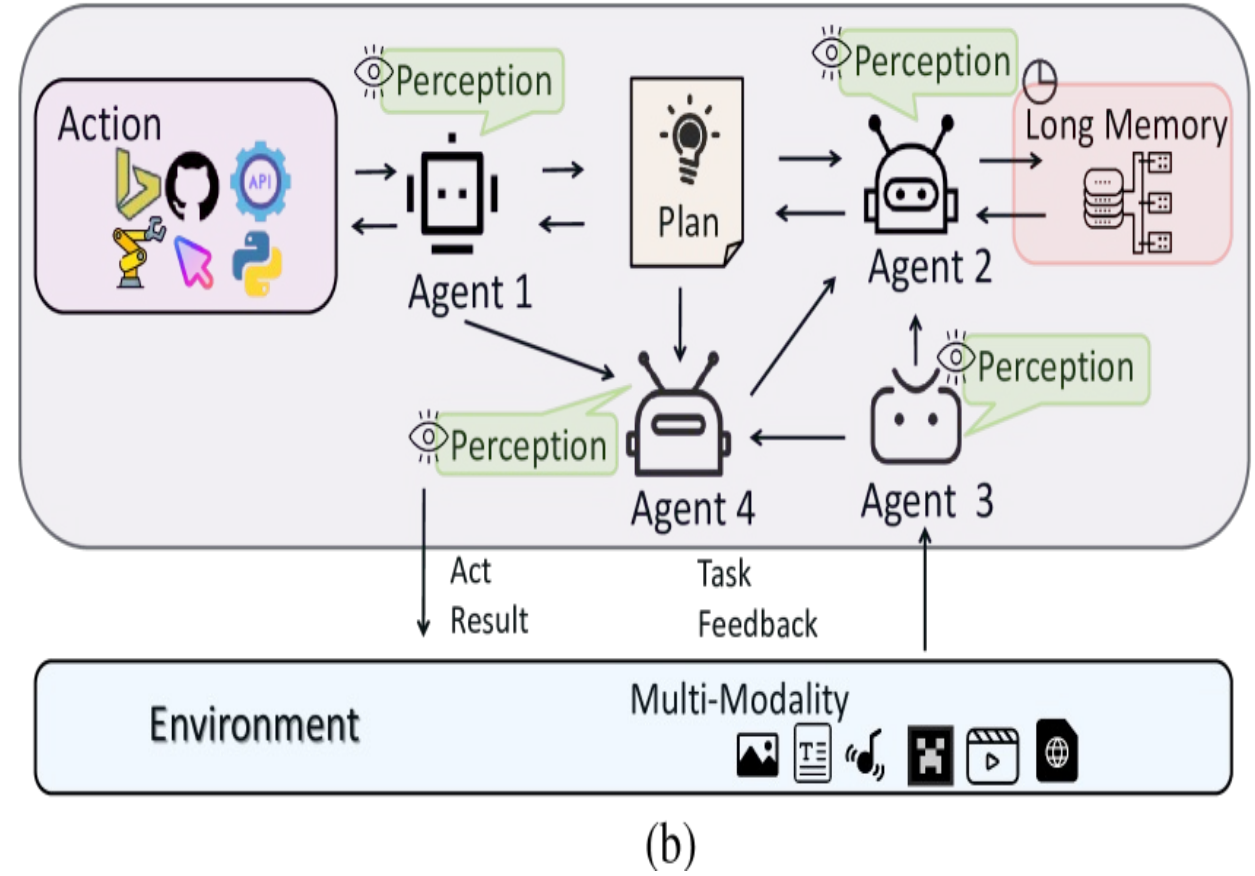
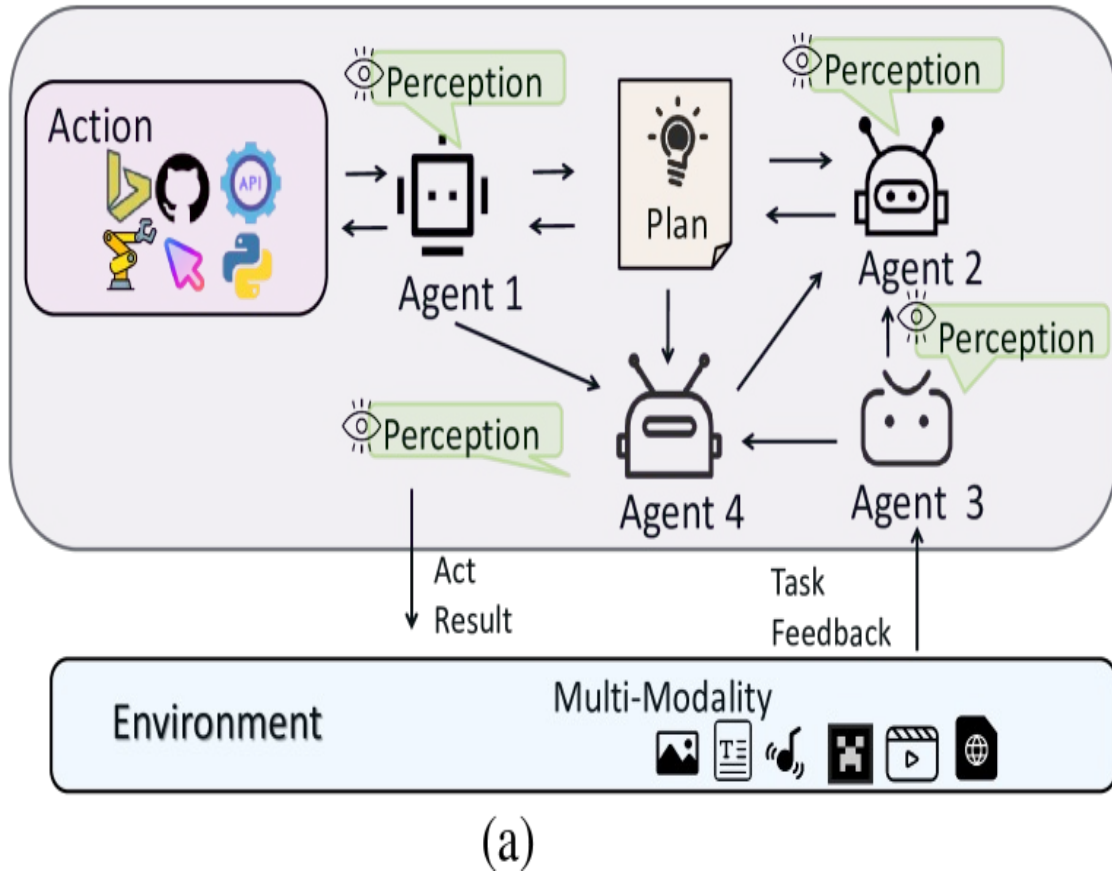


(c)



(d)

Large Multimodal Agents (LMA)



Logical Agents

Logical Agents

Knowledge-based Agents

KB Agents

Knowledge-based Agent (KB Agent)

function **KB-AGENT**(*percept*) **returns** an *action*

persistent: *KB*, a knowledge base

t, a counter, initially 0, indicating time

TELL(*KB*, **MAKE-PERCEPT-SENTENCE**(*percept*, *t*))

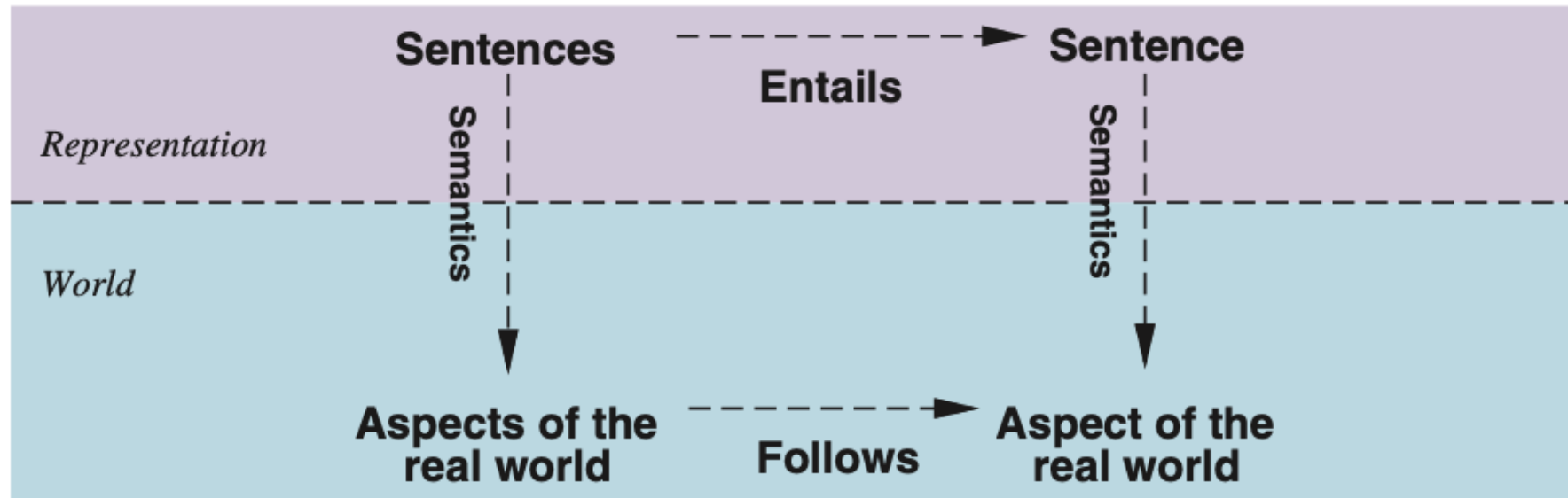
action \leftarrow **ASK**(*KB*, **MAKE-ACTION-QUERY**(*t*))

TELL(*KB*, **MAKE-ACTION-SENTENCE**(*action*, *t*))

t \leftarrow *t* + 1

return *action*

Sentences are physical configurations of the agent



Reasoning is a process of constructing new physical configurations from old ones

Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

A BNF (Backus–Naur Form) grammar of sentences in propositional logic

Sentence \rightarrow *AtomicSentence* | *ComplexSentence*

AtomicSentence \rightarrow *True* | *False* | *P* | *Q* | *R* | ...

ComplexSentence \rightarrow (*Sentence*)
| \neg *Sentence*
| *Sentence* \wedge *Sentence*
| *Sentence* \vee *Sentence*
| *Sentence* \Rightarrow *Sentence*
| *Sentence* \Leftrightarrow *Sentence*

OPERATOR PRECEDENCE : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Truth Tables (TT) for the Five Logical Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

A Truth Table constructed for the knowledge base given in the text

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>

A Truth-Table (TT) enumeration algorithm for deciding propositional entailment

```
function TT-ENTAILS?( $KB, \alpha$ ) returns true or false  
  inputs:  $KB$ , the knowledge base, a sentence in propositional logic  
            $\alpha$ , the query, a sentence in propositional logic  
  
   $symbols \leftarrow$  a list of the proposition symbols in  $KB$  and  $\alpha$   
  return TT-CHECK-ALL( $KB, \alpha, symbols, \{ \}$ )  
  
function TT-CHECK-ALL( $KB, \alpha, symbols, model$ ) returns true or false  
  if EMPTY?( $symbols$ ) then  
    if PL-TRUE?( $KB, model$ ) then return PL-TRUE?( $\alpha, model$ )  
    else return true // when  $KB$  is false, always return true  
  else  
     $P \leftarrow$  FIRST( $symbols$ )  
     $rest \leftarrow$  REST( $symbols$ )  
    return (TT-CHECK-ALL( $KB, \alpha, rest, model \cup \{P = true\}$ )  
            and  
            TT-CHECK-ALL( $KB, \alpha, rest, model \cup \{P = false\}$ ))
```

Standard Logical Equivalences

The symbols α , β , and γ stand for arbitrary sentences of propositional logic.

$$\begin{aligned}(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\ \neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\ \neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\ \neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge\end{aligned}$$

A grammar for Conjunctive Normal Form (CNF), Horn clauses, and definite clauses

CNFSentence \rightarrow *Clause*₁ $\wedge \dots \wedge$ *Clause*_n

Clause \rightarrow *Literal*₁ $\vee \dots \vee$ *Literal*_m

Fact \rightarrow *Symbol*

Literal \rightarrow *Symbol* | \neg *Symbol*

Symbol \rightarrow *P* | *Q* | *R* | ...

HornClauseForm \rightarrow *DefiniteClauseForm* | *GoalClauseForm*

DefiniteClauseForm \rightarrow *Fact* | (*Symbol*₁ $\wedge \dots \wedge$ *Symbol*_l) \Rightarrow *Symbol*

GoalClauseForm \rightarrow (*Symbol*₁ $\wedge \dots \wedge$ *Symbol*_l) \Rightarrow *False*

A simple resolution algorithm for propositional logic

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
inputs: KB , the knowledge base, a sentence in propositional logic
 α , the query, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
 $new \leftarrow \{ \}$
while *true* **do**
 for each pair of clauses C_i, C_j **in** $clauses$ **do**
 $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
 if $resolvents$ contains the empty clause **then return** *true*
 $new \leftarrow new \cup resolvents$
 if $new \subseteq clauses$ **then return** *false*
 $clauses \leftarrow clauses \cup new$

The forward-chaining algorithm for propositional logic

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*

inputs: KB , the knowledge base, a set of propositional definite clauses

q , the query, a proposition symbol

$count \leftarrow$ a table, where $count[c]$ is initially the number of symbols in clause c 's premise

$inferred \leftarrow$ a table, where $inferred[s]$ is initially *false* for all symbols

$queue \leftarrow$ a queue of symbols, initially symbols known to be true in KB

while $queue$ is not empty **do**

$p \leftarrow$ POP($queue$)

if $p = q$ **then return** *true*

if $inferred[p] = false$ **then**

$inferred[p] \leftarrow true$

for each clause c in KB where p is in c .PREMISE **do**

 decrement $count[c]$

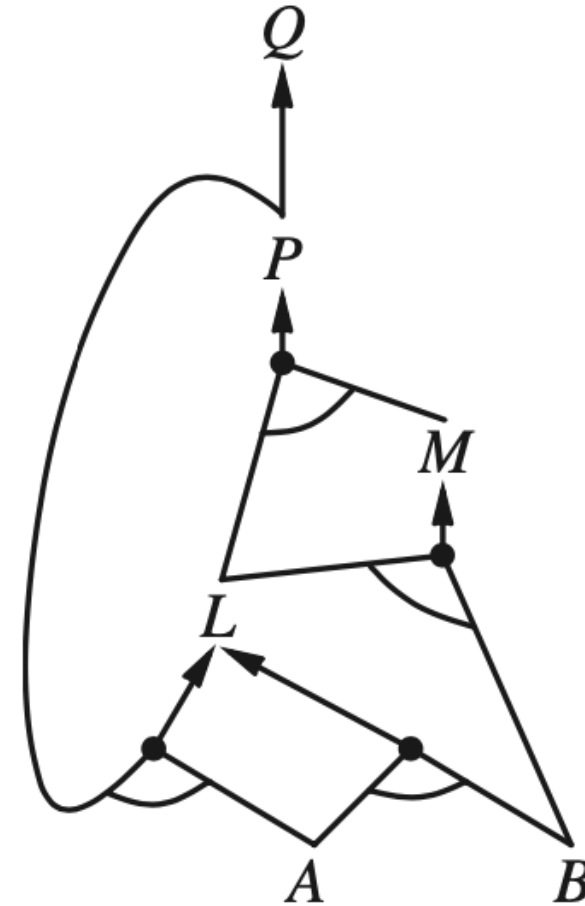
if $count[c] = 0$ **then** add c .CONCLUSION to $queue$

return *false*

A set of Horn clauses

- $P \Rightarrow Q$
- $L \wedge M \Rightarrow P$
- $B \wedge L \Rightarrow M$
- $A \wedge P \Rightarrow L$
- $A \wedge B \Rightarrow L$
- A
- B

(a)



(b)

The corresponding AND-OR graph

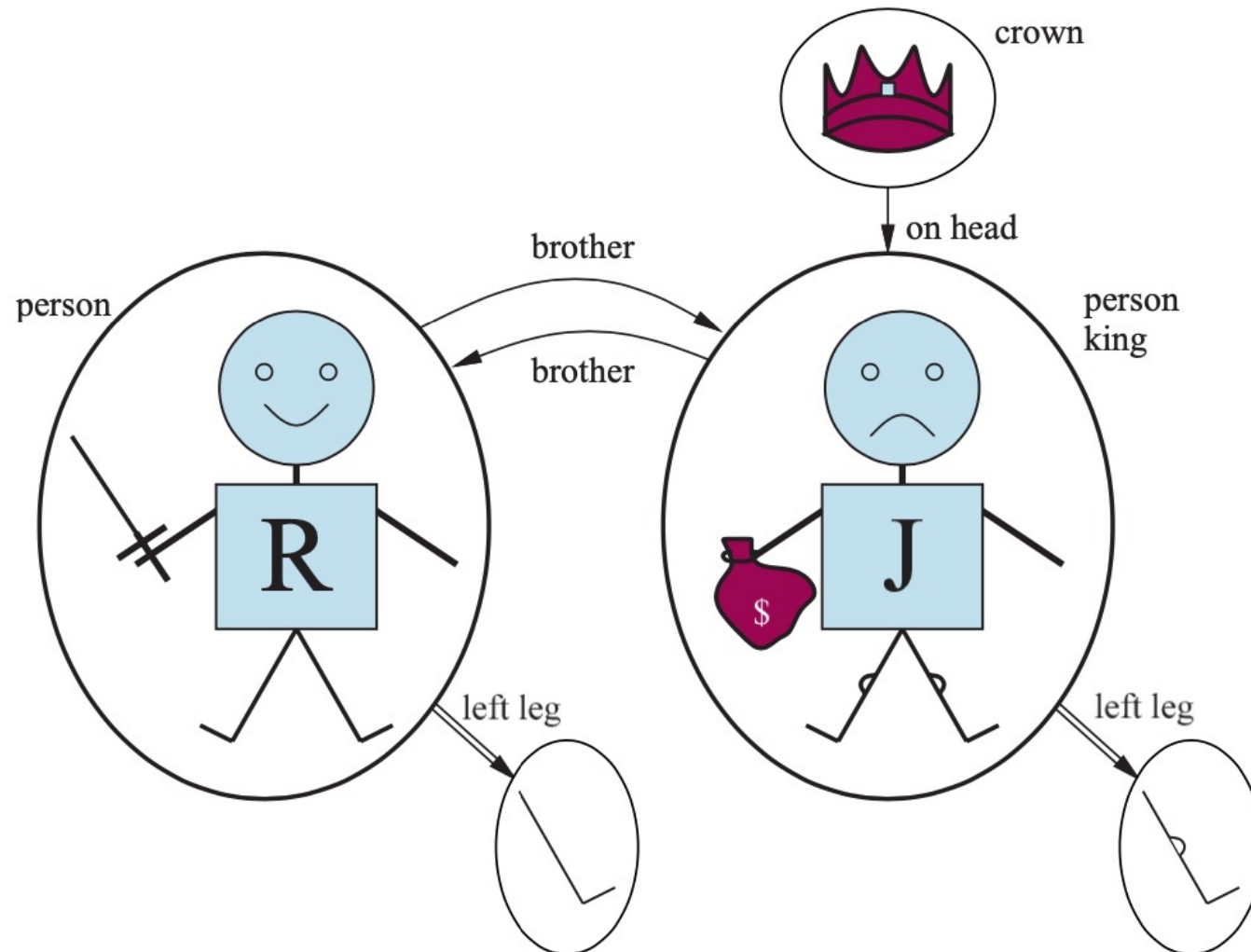
First-Order Logic

Formal languages and their ontological and epistemological commitments

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

A model containing five objects

two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).



The syntax of first-order logic with equality

Sentence → *AtomicSentence* | *ComplexSentence*

AtomicSentence → *Predicate* | *Predicate(Term, ...)* | *Term = Term*

ComplexSentence → (*Sentence*)
| \neg *Sentence*
| *Sentence* \wedge *Sentence*
| *Sentence* \vee *Sentence*
| *Sentence* \Rightarrow *Sentence*
| *Sentence* \Leftrightarrow *Sentence*
| *Quantifier Variable, ... Sentence*

Term → *Function(Term, ...)*
| *Constant*
| *Variable*

Quantifier → \forall | \exists

Constant → *A* | *X*₁ | *John* | ...

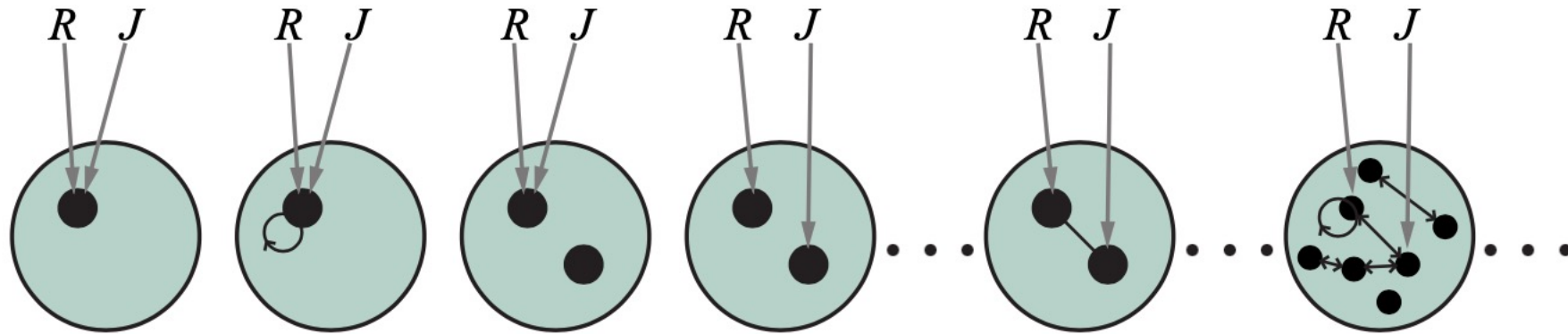
Variable → *a* | *x* | *s* | ...

Predicate → *True* | *False* | *After* | *Loves* | *Raining* | ...

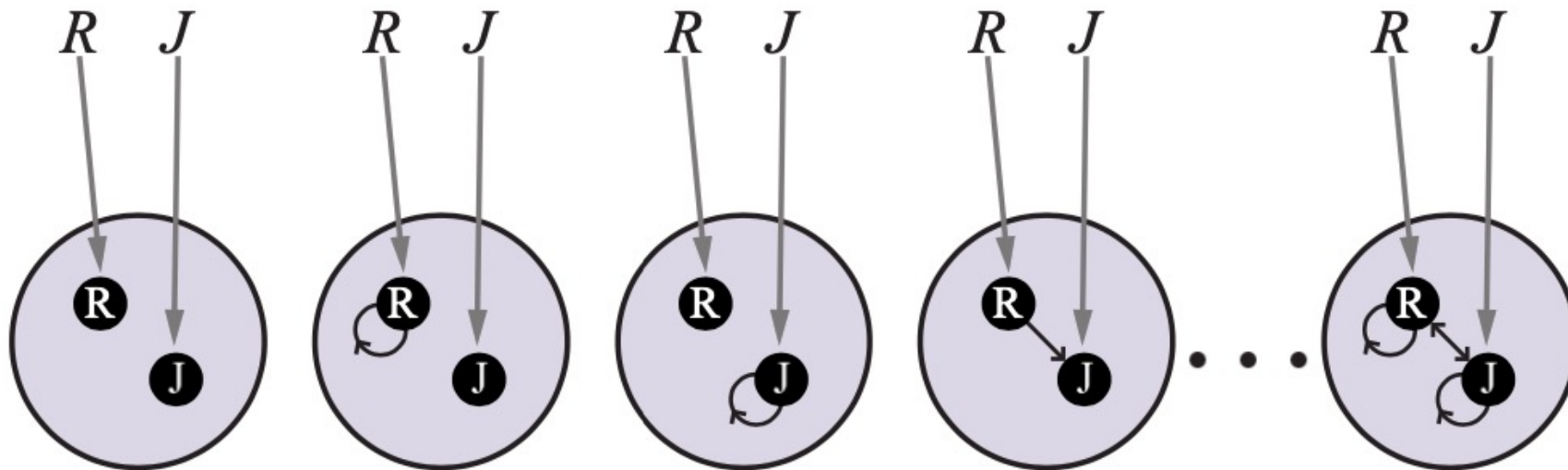
Function → *Mother* | *LeftLeg* | ...

OPERATOR PRECEDENCE : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

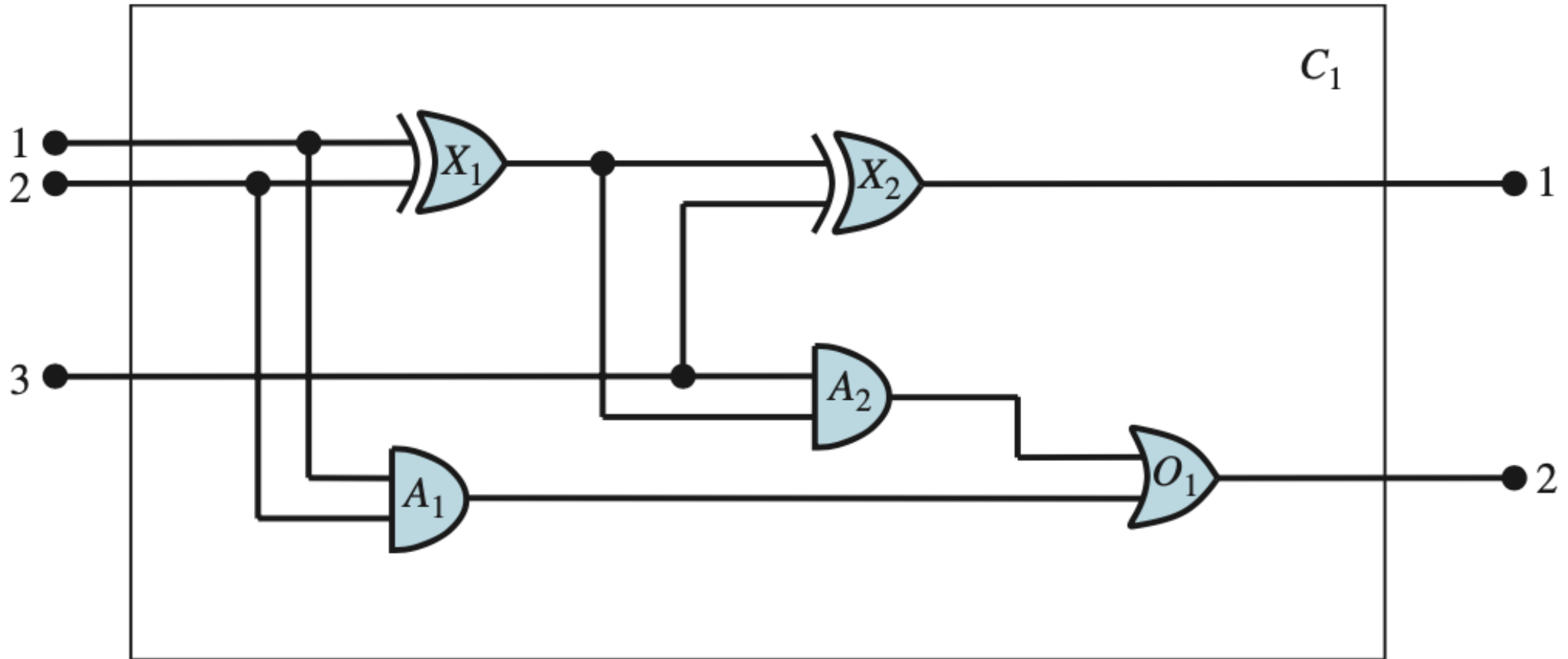
Some members of the set of all models for a language with two constant symbols, R and J , and one binary relation symbol



Some members of the set of all models for a language with two constant symbols, R and J , and one binary relation symbol, under database semantics



A digital circuit C1, purporting to be a one-bit full adder.



Inference in First-Order Logic

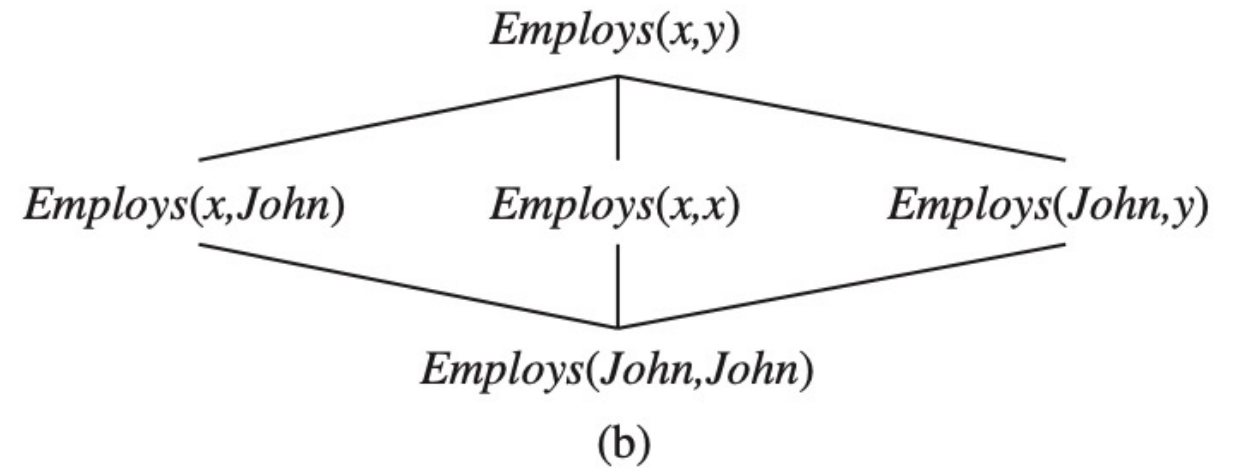
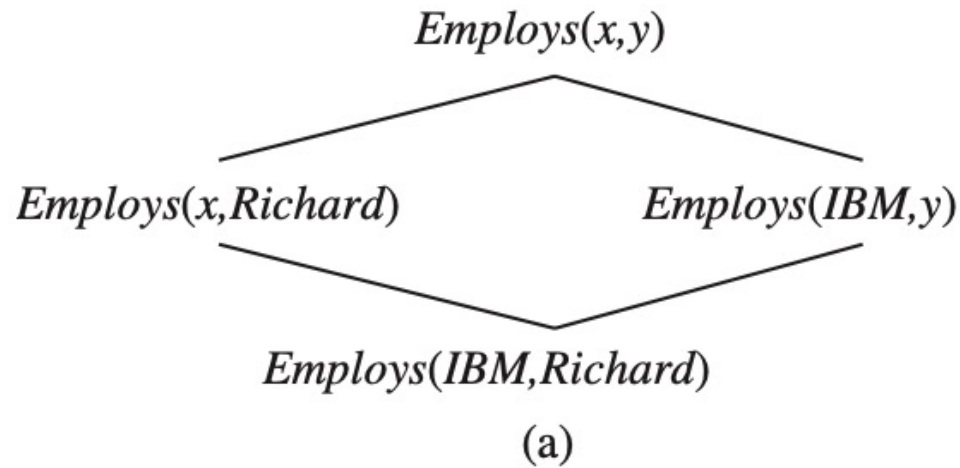
The unification algorithm

function UNIFY($x, y, \theta = \text{empty}$) **returns** a substitution to make x and y identical, or *failure*
if $\theta = \text{failure}$ **then return** *failure*
else if $x = y$ **then return** θ
else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
else if COMPOUND?(x) **and** COMPOUND?(y) **then**
 return UNIFY(ARGS(x), ARGS(y), UNIFY(OP(x), OP(y), θ))
else if LIST?(x) **and** LIST?(y) **then**
 return UNIFY(REST(x), REST(y), UNIFY(FIRST(x), FIRST(y), θ))
else return *failure*

function UNIFY-VAR(var, x, θ) **returns** a substitution
if $\{var/val\} \in \theta$ for some val **then return** UNIFY(val, x, θ)
else if $\{x/val\} \in \theta$ for some val **then return** UNIFY(var, val, θ)
else if OCCUR-CHECK?(var, x) **then return** *failure*
else return add $\{var/x\}$ to θ

The subsumption lattice whose lowest node is **Employs** **(IBM , Richard)**

The subsumption lattice for the sentence **Employs**
(John, John)



A conceptually straightforward, but inefficient, forward-chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

inputs: KB , the knowledge base, a set of first-order definite clauses

α , the query, an atomic sentence

while *true* **do**

$new \leftarrow \{ \}$ // *The set of new sentences inferred on each iteration*

for each *rule* **in** KB **do**

$(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}(\text{rule})$

for each θ such that $\text{SUBST}(\theta, p_1 \wedge \dots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \dots \wedge p'_n)$

for some p'_1, \dots, p'_n in KB

$q' \leftarrow \text{SUBST}(\theta, q)$

if q' does not unify with some sentence already in KB or *new* **then**

add q' to *new*

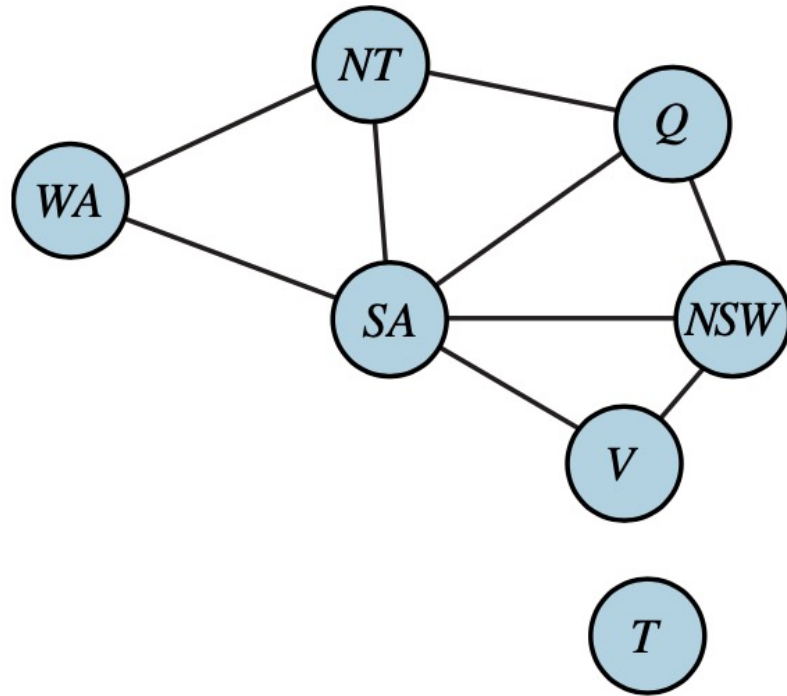
$\phi \leftarrow \text{UNIFY}(q', \alpha)$

if ϕ is not *failure* **then return** ϕ

if $new = \{ \}$ **then return** *false*

add *new* to KB

Constraint graph for coloring the map of Australia



(a)

$$\begin{aligned} & Diff(wa, nt) \wedge Diff(wa, sa) \wedge \\ & Diff(nt, q) \wedge Diff(nt, sa) \wedge \\ & Diff(q, nsw) \wedge Diff(q, sa) \wedge \\ & Diff(nsw, v) \wedge Diff(nsw, sa) \wedge \\ & Diff(v, sa) \Rightarrow Colorable() \end{aligned}$$

$$\begin{aligned} & Diff(Blue, Red) \quad Diff(Blue, Green) \\ & Diff(Green, Red) \quad Diff(Green, Blue) \\ & Diff(Red, Blue) \quad Diff(Red, Green) \end{aligned}$$

(b)

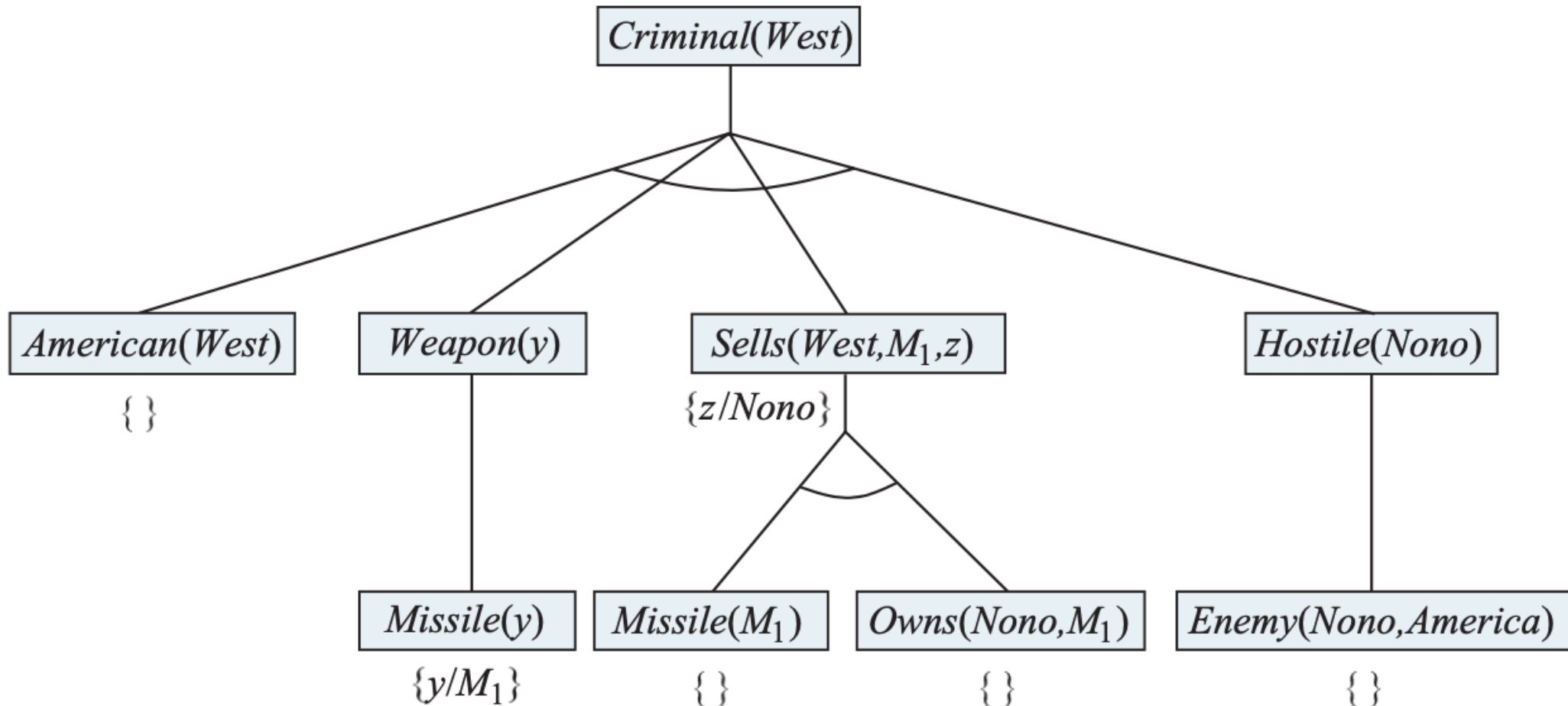
A simple backward-chaining algorithm for first-order knowledge bases

function FOL-BC-ASK($KB, query$) **returns** a generator of substitutions
return FOL-BC-OR($KB, query, \{ \}$)

function FOL-BC-OR($KB, goal, \theta$) **returns** a substitution
for each $rule$ in FETCH-RULES-FOR-GOAL($KB, goal$) **do**
 $(lhs \Rightarrow rhs) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
 for each θ' in FOL-BC-AND($KB, lhs, UNIFY(rhs, goal, \theta)$) **do**
 yield θ'

function FOL-BC-AND($KB, goals, \theta$) **returns** a substitution
if $\theta = failure$ **then return**
else if LENGTH($goals$) = 0 **then yield** θ
else
 $first, rest \leftarrow$ FIRST($goals$), REST($goals$)
 for each θ' in FOL-BC-OR($KB, SUBST(\theta, first), \theta$) **do**
 for each θ'' in FOL-BC-AND($KB, rest, \theta'$) **do**
 yield θ''

Proof tree constructed by backward chaining to prove that West is a criminal



Pseudocode representing the result of compiling the Append predicate

procedure APPEND($ax, y, az, continuation$)

$trail \leftarrow$ GLOBAL-TRAIL-POINTER()

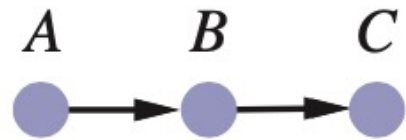
if $ax = []$ and UNIFY(y, az) **then** CALL($continuation$)

RESET-TRAIL($trail$)

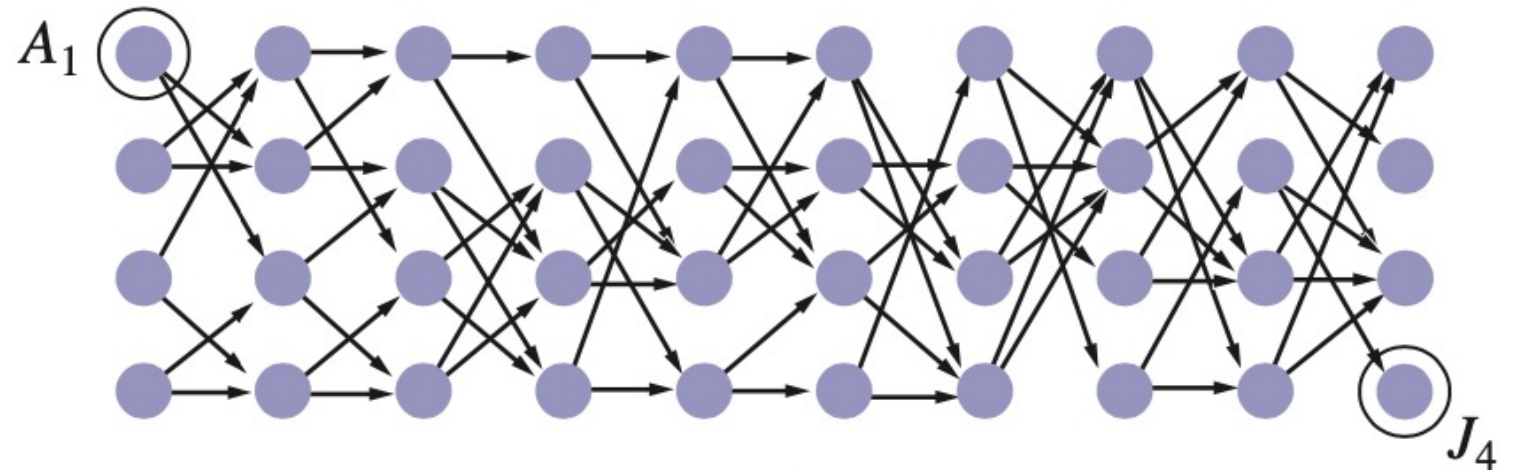
$a, x, z \leftarrow$ NEW-VARIABLE(), NEW-VARIABLE(), NEW-VARIABLE()

if UNIFY($ax, [a] + x$) and UNIFY($az, [a | z]$) **then** APPEND($x, y, z, continuation$)

Finding a path from A to C can lead Prolog into an infinite loop.

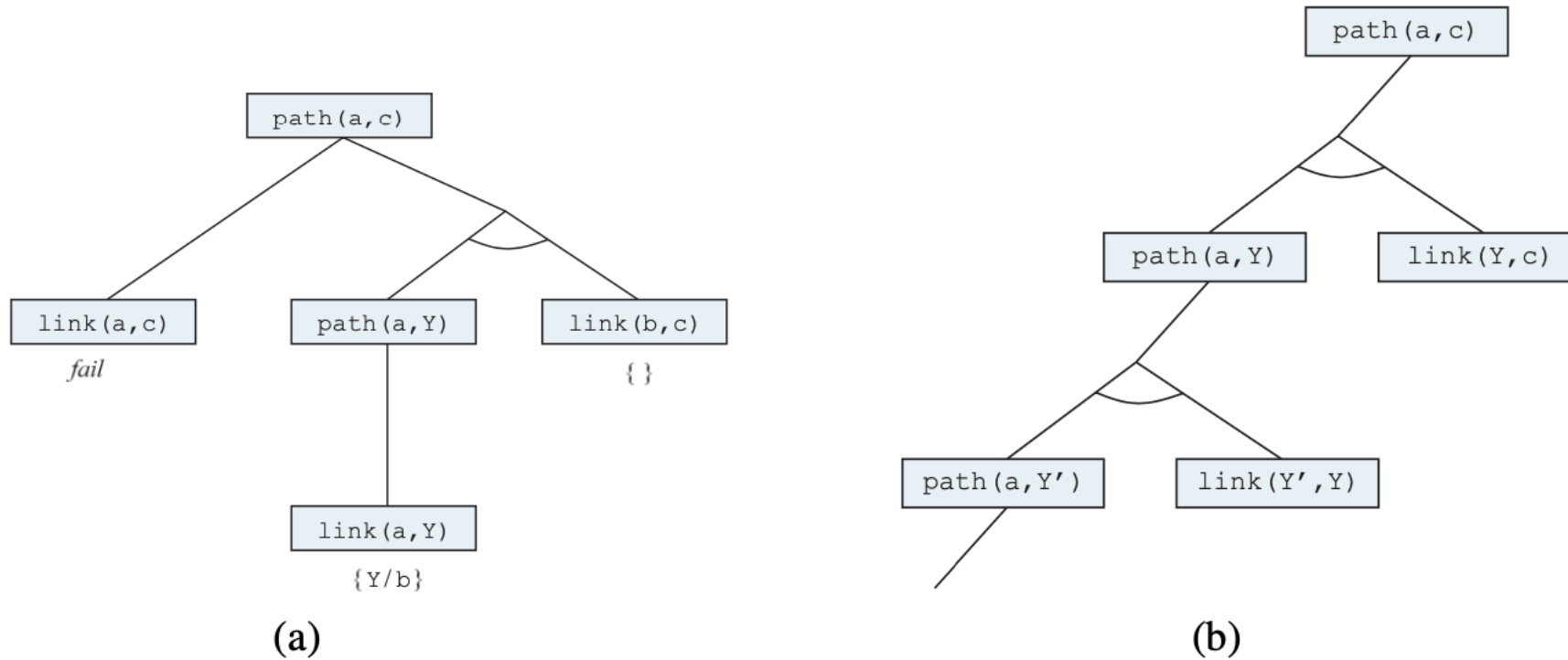


(a)



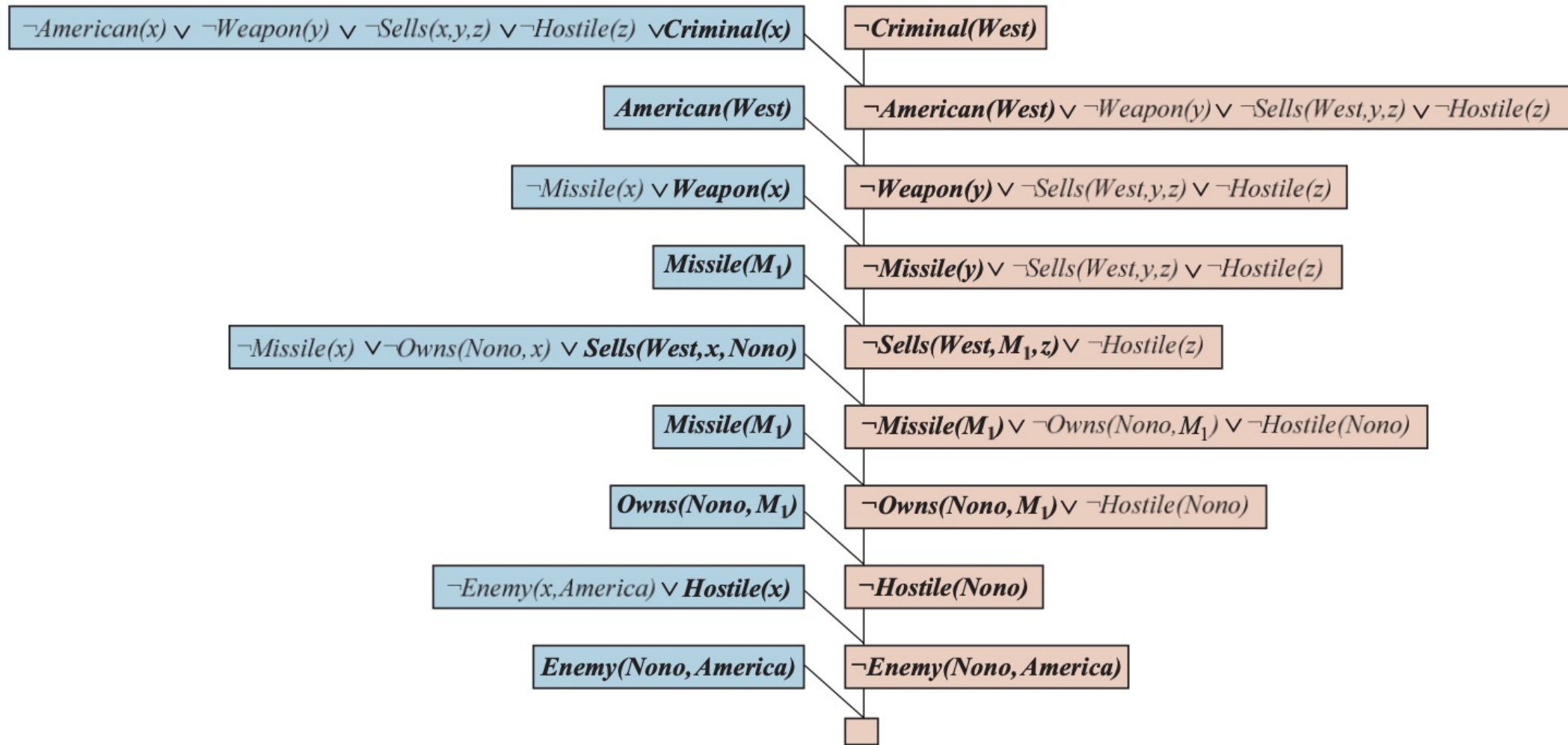
(b)

Proof that a path exists from A to C.

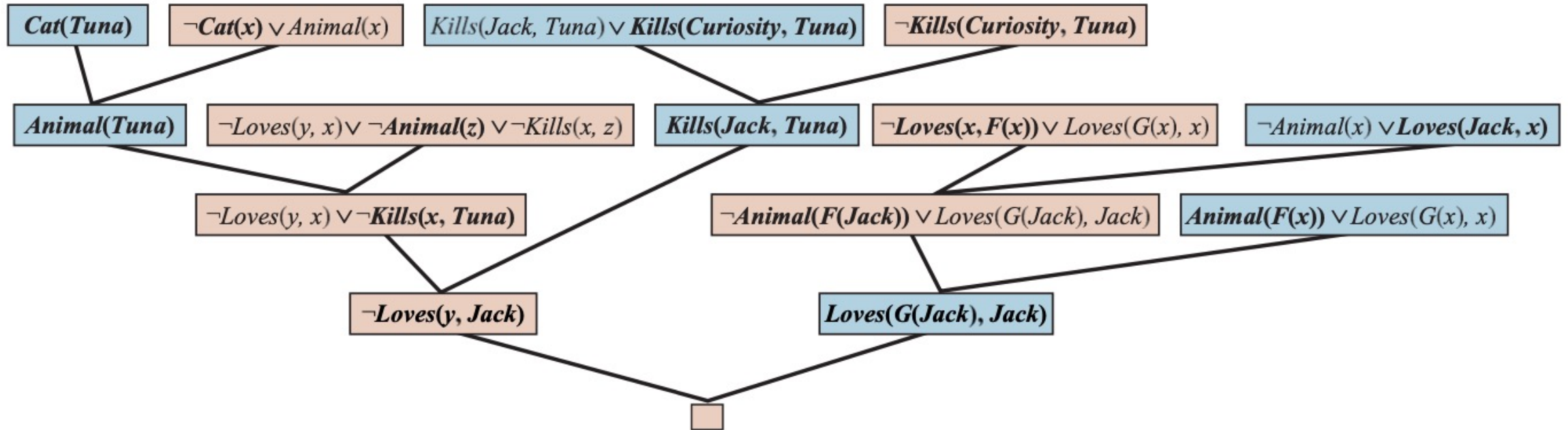


Infinite proof tree generated when the clauses are in the “wrong” order

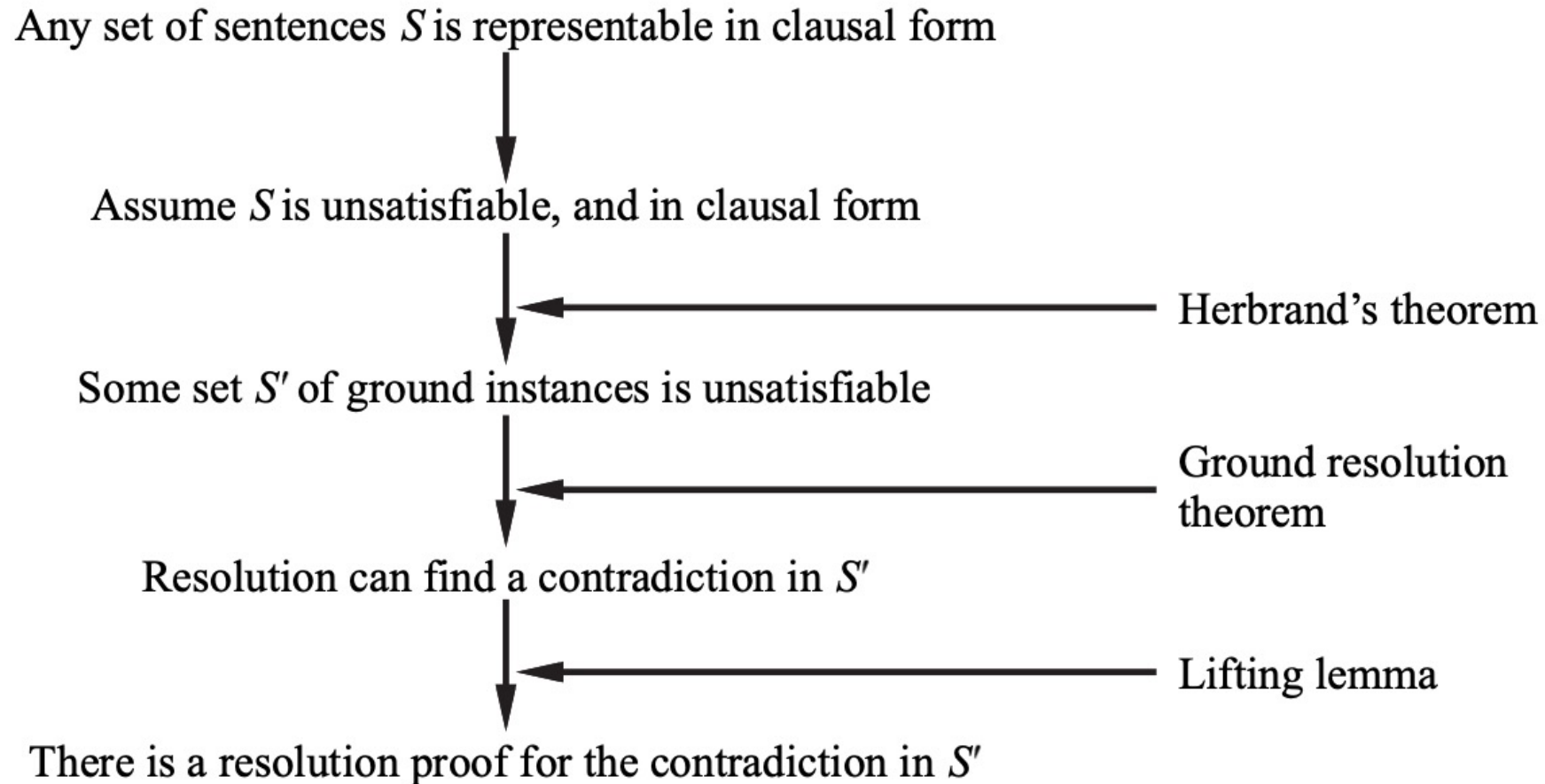
A resolution proof that West is a criminal



A resolution proof that Curiosity killed the cat

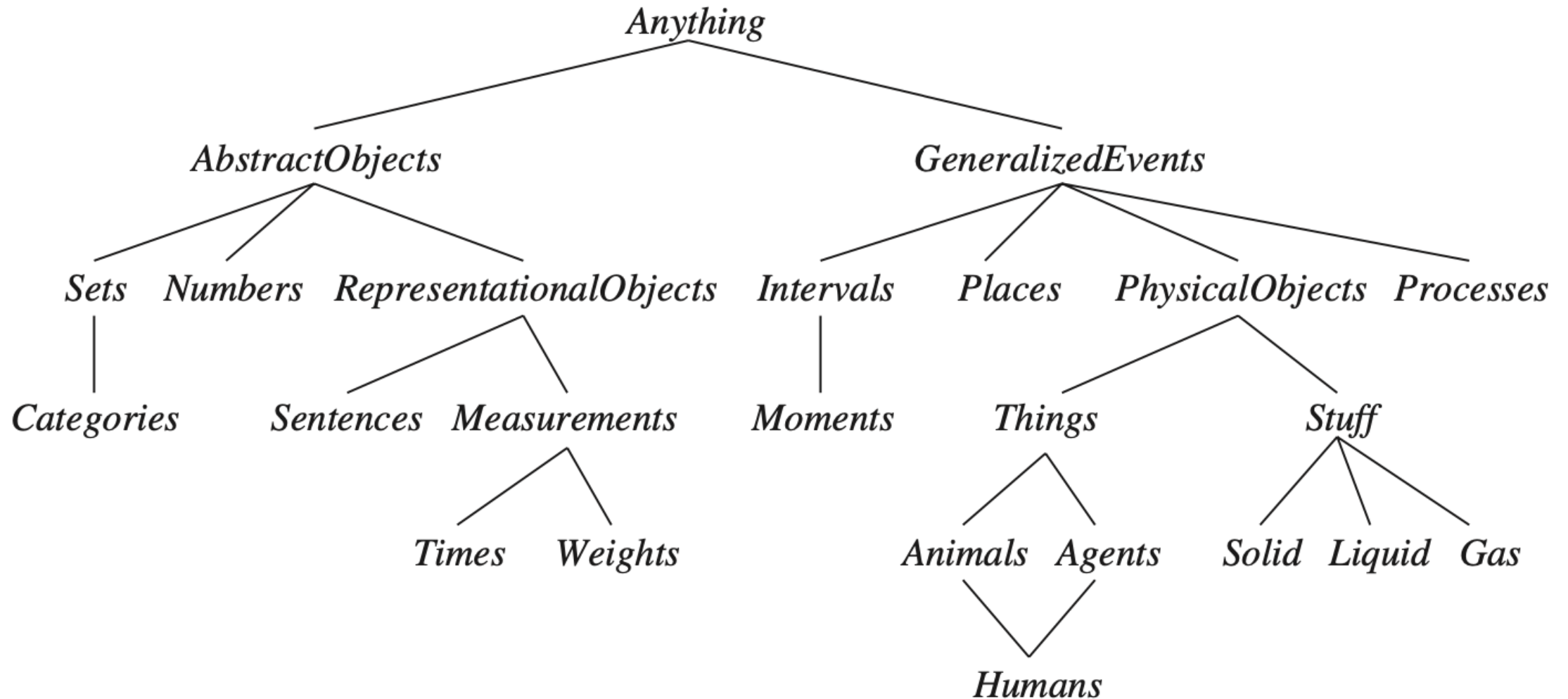


Structure of a completeness proof for resolution

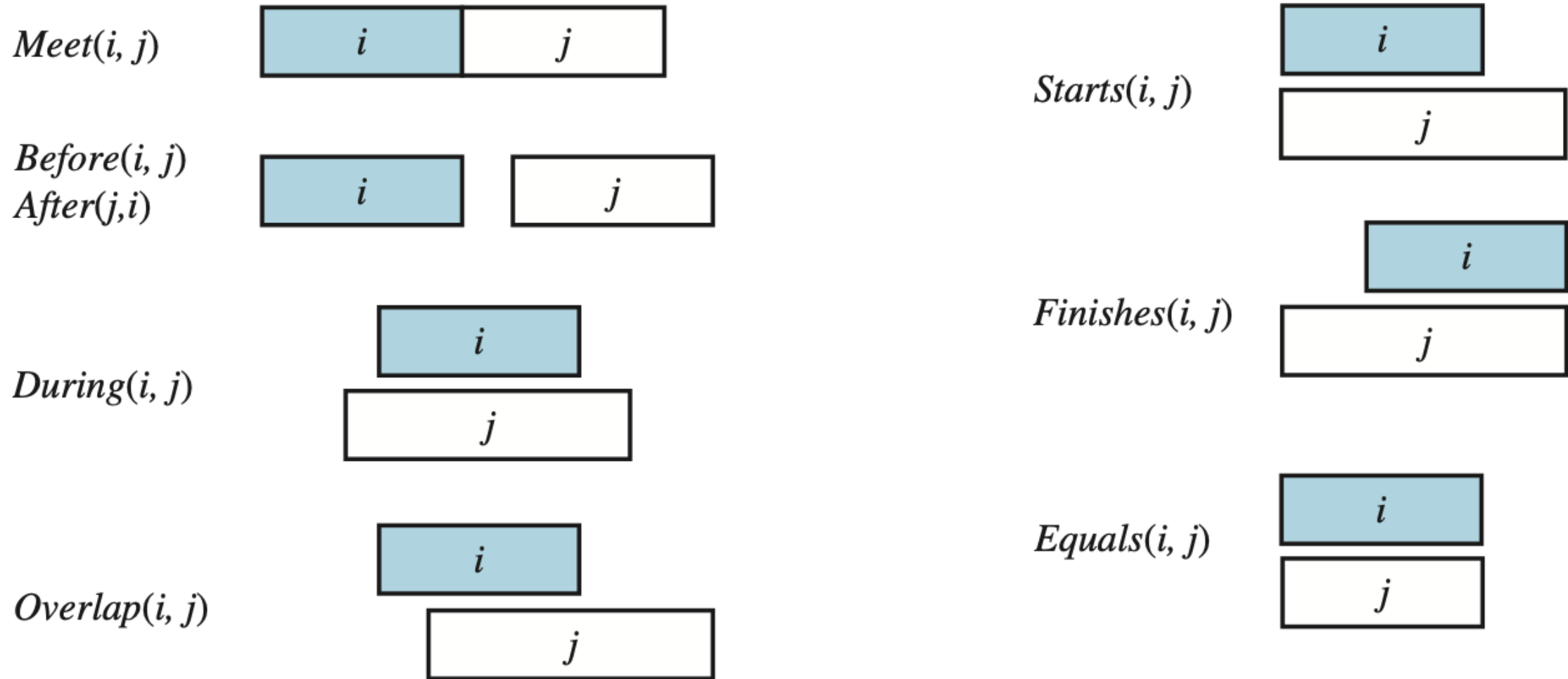


Knowledge Representation

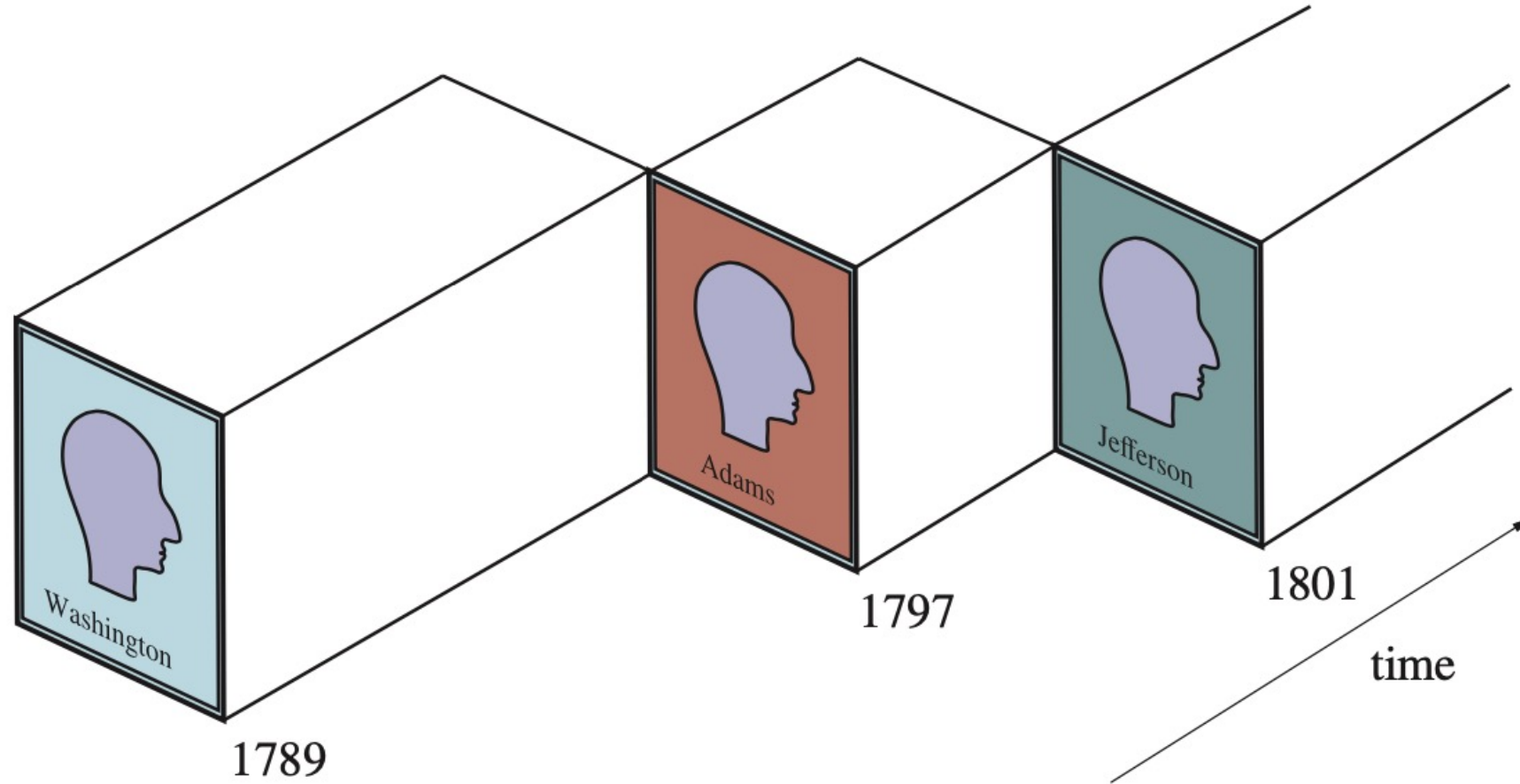
The Upper Ontology of the World



Predicates on time intervals

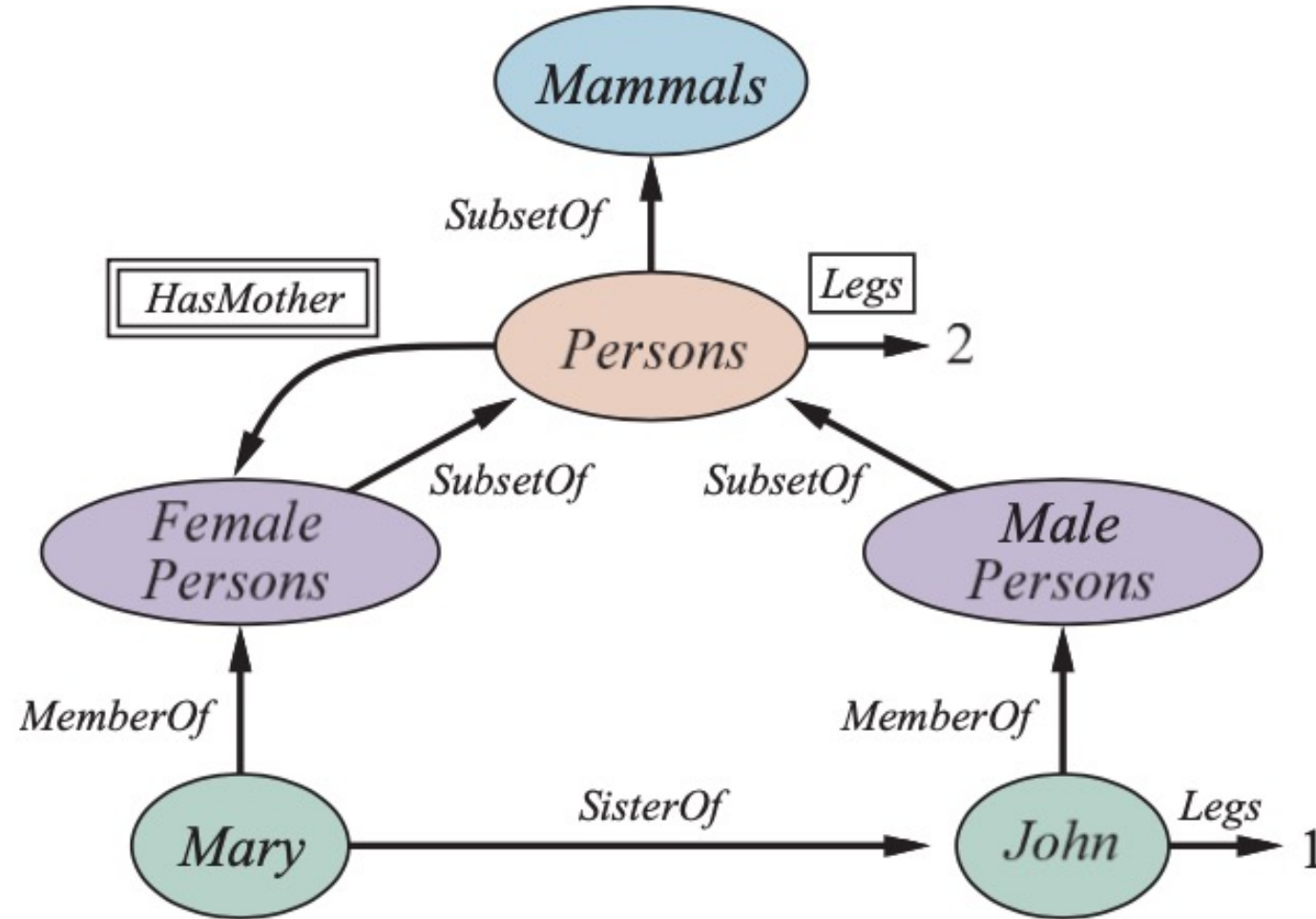


A schematic view of the object President (USA) for the early years



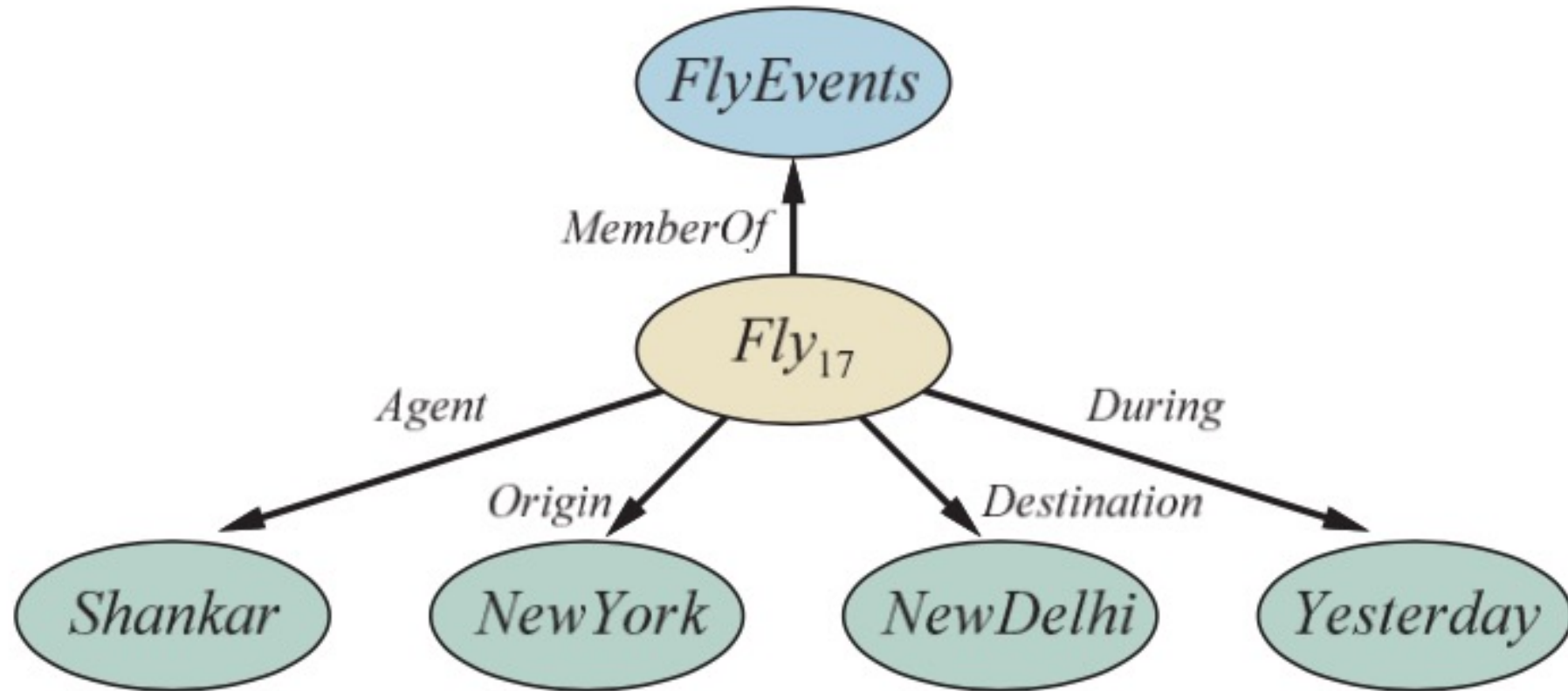
A semantic network

with four objects (John, Mary, 1, and 2) and four categories Relations are denoted by labeled links



Semantic network

Representation of the logical assertion
Fly (Shankar, NewYork, NewDelhi, Yesterday)



The syntax of descriptions in a subset of the CLASSIC language.

Concept → **Thing** | *ConceptName*
| **And**(*Concept*, ...)
| **All**(*RoleName*, *Concept*)
| **AtLeast**(*Integer*, *RoleName*)
| **AtMost**(*Integer*, *RoleName*)
| **Fills**(*RoleName*, *IndividualName*, ...)
| **SameAs**(*Path*, *Path*)
| **OneOf**(*IndividualName*, ...)

Path → [*RoleName*, ...]

ConceptName → *Adult* | *Female* | *Male* | ...

RoleName → *Spouse* | *Daughter* | *Son* | ...

Knowledge Graph (KG)

Knowledge Graph (KG)

- **Knowledge Graph (KG)**

- A knowledge graph is a multi-relational graph composed of **entities** and **relations**, which are regarded as **nodes** and different types of **edges**, respectively (Ji et al., 2021).
- Represents knowledge as **concepts (entities)** and their **relationships (Facts)**
- **Triple** of facts
 - *SPO: (subject, predicate, object)*
 - *HRT: (head, relation, tail)*
- **Common Knowledge Graph: DBpedia, YAGO, Wikidata**

Knowledge Graph, Facts, Triple, Embedding

- G
 - Knowledge graph
- F
 - Set of facts
- (h, r, t)
 - Triple of head, relation, and tail
- $(\mathbf{h}, \mathbf{r}, \mathbf{t})$
 - Embedding of head, relation, and tail

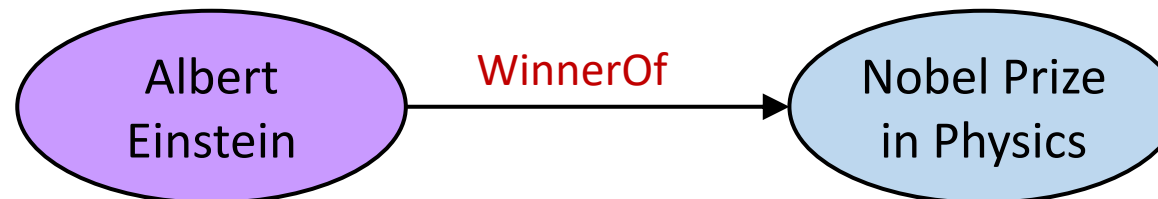
Knowledge Representation

Factual Triple and Knowledge Graph

- Albert Einstein, **winner of** the 1921 Nobel prize in physics
- The Nobel Prize in Physics 1921 was **awarded to** Albert Einstein "for his services to Theoretical Physics, and especially for his discovery of the law of the photoelectric effect."

Triple (Albert Einstein, **WinnerOf**, Nobel Prize in Physics)

**Knowledge
Graph**



Factual Triples in Knowledge Base

(h, r, t)

(Albert Einstein, **BornIn**, German Empire)

(Albert Einstein, **SonOf**, Hermann Einstein)

(Albert Einstein, **GraduateFrom**, University of Zurich)

(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)

(Albert Einstein, **ExpertIn**, Physics)

(Nobel Prize in Physics, **AwardIn**, Physics)

(The theory of relativity, **TheoryOf**, Physics)

(Albert Einstein, **SupervisedBy**, Alfred Kleiner)

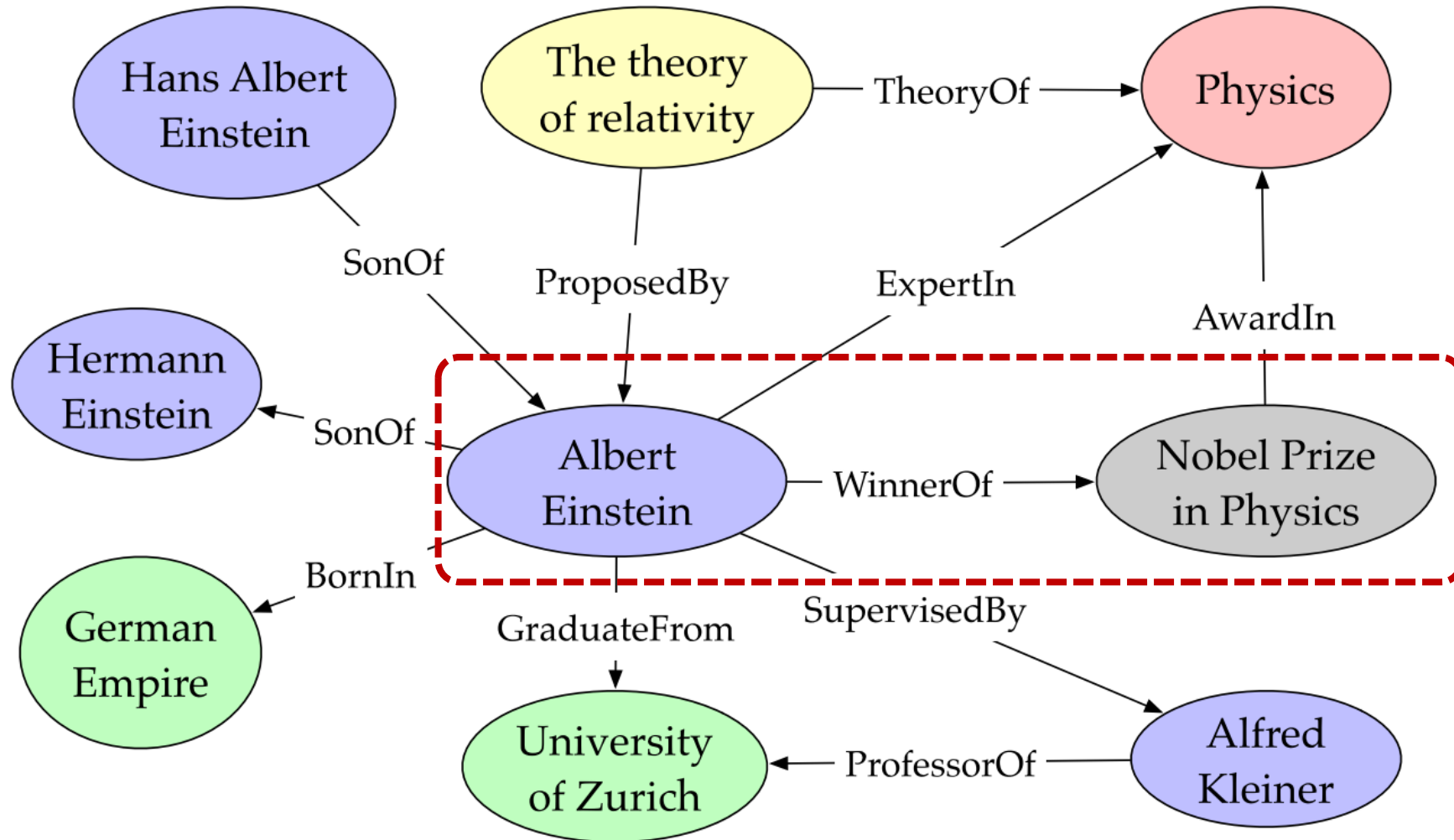
(Alfred Kleiner, **ProfessorOf**, University of Zurich)

(The theory of relativity, **ProposedBy**, Albert Einstein)

(Hans Albert Einstein, **SonOf**, Albert Einstein)

Entities and Relations in Knowledge Graph

(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)



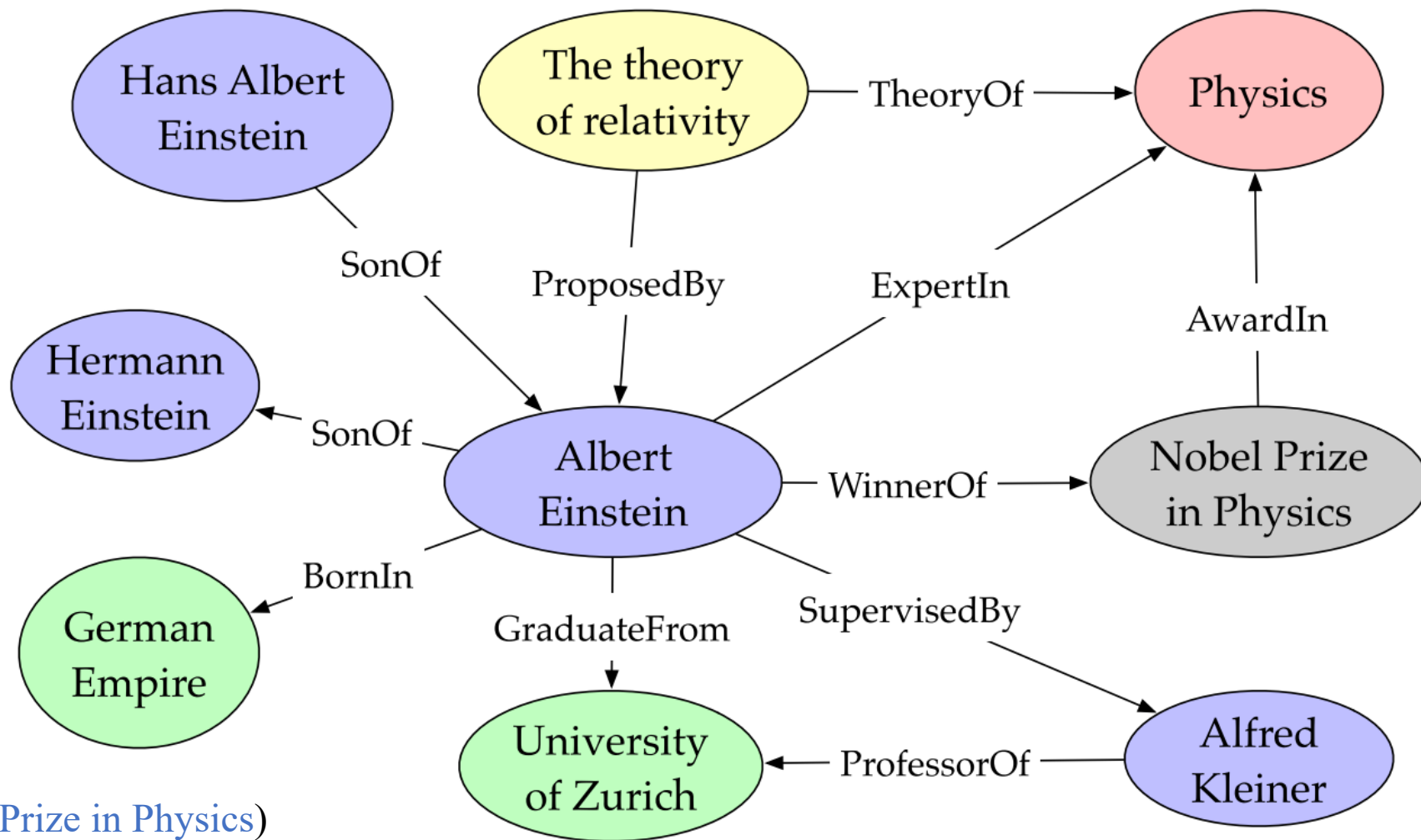
knowledge base and knowledge graph

Factual triples in knowledge base

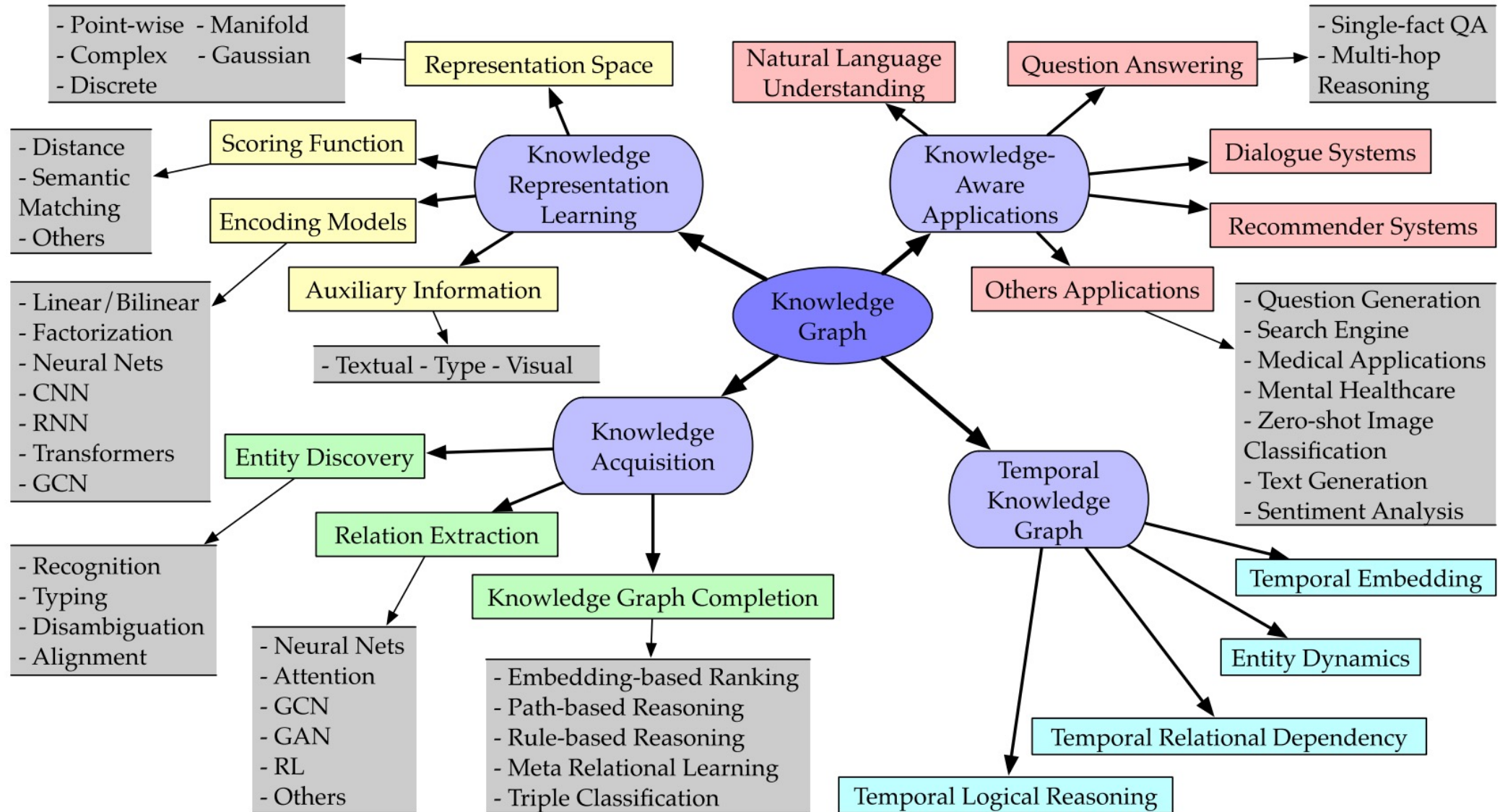
- (Albert Einstein, **BornIn**, German Empire)
- (Albert Einstein, **SonOf**, Hermann Einstein)
- (Albert Einstein, **GraduateFrom**, University of Zurich)
- (Albert Einstein, **WinnerOf**, Nobel Prize in Physics)
- (Albert Einstein, **ExpertIn**, Physics)
- (Nobel Prize in Physics, **AwardIn**, Physics)
- (The theory of relativity, **TheoryOf**, Physics)
- (Albert Einstein, **SupervisedBy**, Alfred Kleiner)
- (Alfred Kleiner, **ProfessorOf**, University of Zurich)
- (The theory of relativity, **ProposedBy**, Albert Einstein)
- (Hans Albert Einstein, **SonOf**, Albert Einstein)

(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)

Entities and relations in knowledge graph



Categorization of Research on Knowledge Graphs



Source: Ji, S., Pan, S., Cambria, E., Marttinen, P., & Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications.

IEEE Transactions on Neural Networks and Learning Systems.

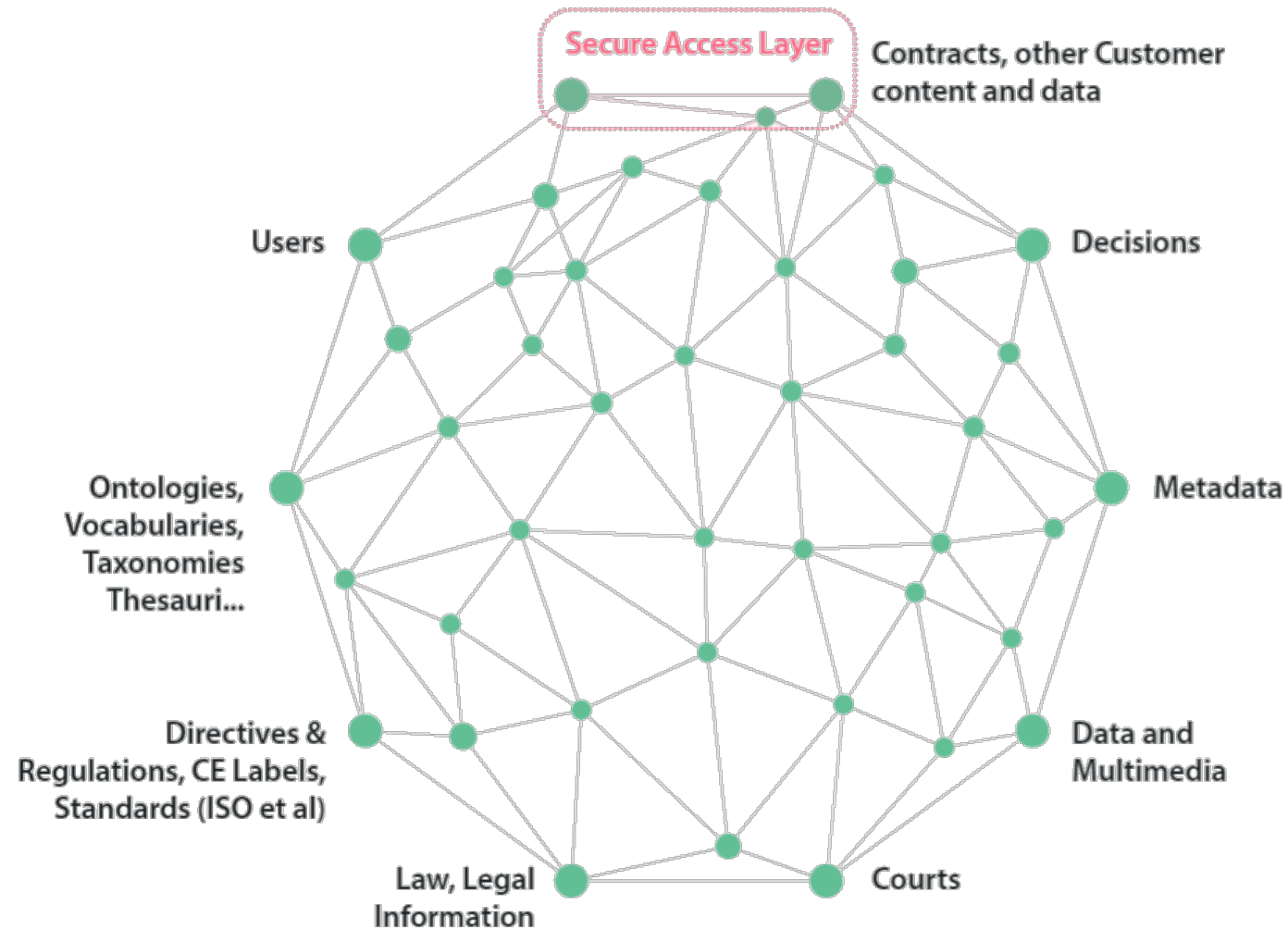
Knowledge Graph Completion (KGC) Datasets

Knowledge Graph Completion (KGC) Dataset	#Entity	#Relation	#Train	#Valid	#Test	Reference
WN18RR	40,943	11	86,835	3,034	3,134	Toutanova & Chen (2015); Zhang et al. (2020)
FB15k-237	14,541	237	272,115	17,535	20,466	Dettmers et al. (2018); Zhang et al. (2020)
YAGO3-10	123,182	37	1,079,040	5,000	5,000	Mahdisoltani et al. (2015); Zhang et al. (2020)

Domain-Specific Knowledge Graph

- **Domain-Specific Knowledge Graph**
 - **PubMed Knowledge Graph (PKG)**
 - **Extracting biological entities from 29 million PubMed abstracts**
 - **Lynx: Legal Knowledge Graph for Multilingual Compliance Services**
 - **Legal Knowledge Graph (LKG) integrates and links heterogeneous compliance data sources including legislation, case law, standards and other private contracts.**

Lynx: Legal Knowledge Graph for Multilingual Compliance Services



Automated Planning

A PDDL description of an air cargo transportation planning problem

Init($At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
 $\wedge Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
 $\wedge Airport(JFK) \wedge Airport(SFO)$)

Goal($At(C_1, JFK) \wedge At(C_2, SFO)$)

Action(*Load*(c, p, a),

PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$

EFFECT: $\neg At(c, a) \wedge In(c, p)$)

Action(*Unload*(c, p, a),

PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$

EFFECT: $At(c, a) \wedge \neg In(c, p)$)

Action(*Fly*($p, from, to$),

PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$

EFFECT: $\neg At(p, from) \wedge At(p, to)$)

The simple spare tire problem

Init(*Tire*(*Flat*) \wedge *Tire*(*Spare*) \wedge *At*(*Flat*, *Axle*) \wedge *At*(*Spare*, *Trunk*))

Goal(*At*(*Spare*, *Axle*))

Action(*Remove*(*obj*, *loc*),

PRECOND: *At*(*obj*, *loc*)

EFFECT: \neg *At*(*obj*, *loc*) \wedge *At*(*obj*, *Ground*))

Action(*PutOn*(*t*, *Axle*),

PRECOND: *Tire*(*t*) \wedge *At*(*t*, *Ground*) \wedge \neg *At*(*Flat*, *Axle*) \wedge \neg *At*(*Spare*, *Axle*)

EFFECT: \neg *At*(*t*, *Ground*) \wedge *At*(*t*, *Axle*))

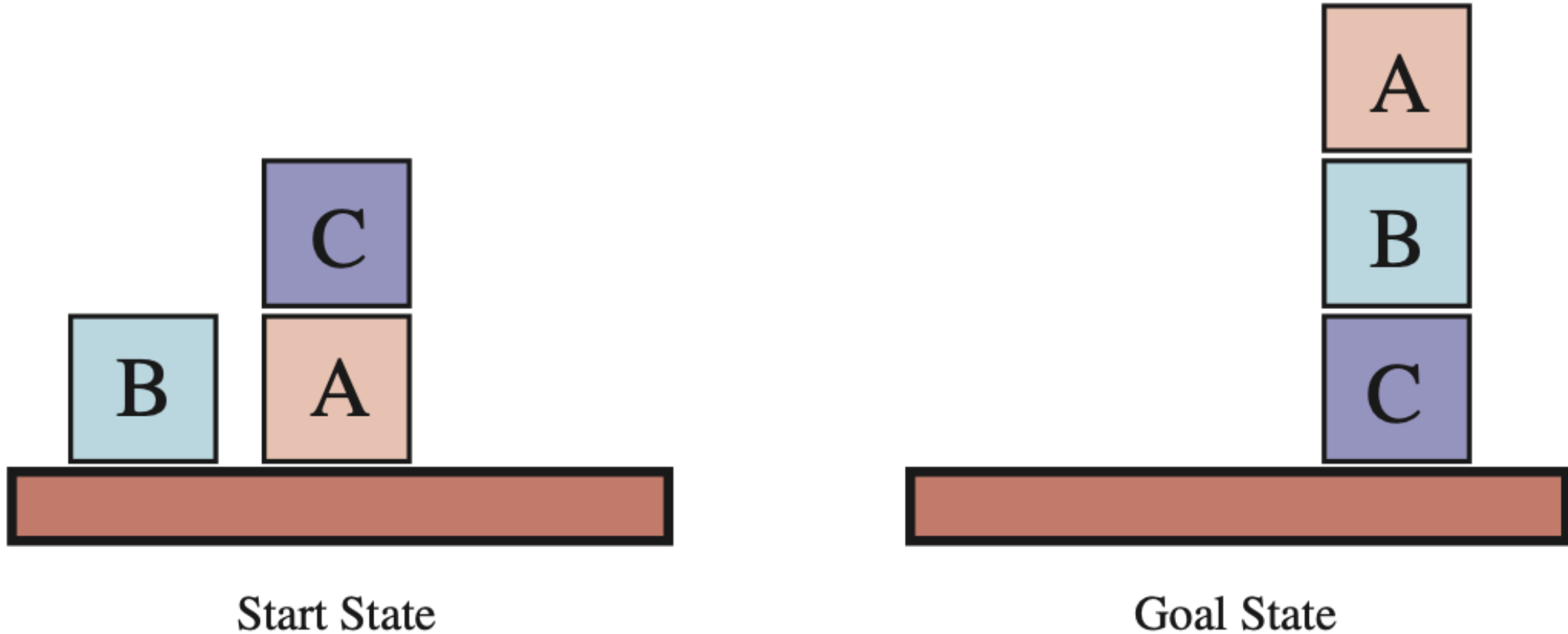
Action(*LeaveOvernight*,

PRECOND:

EFFECT: \neg *At*(*Spare*, *Ground*) \wedge \neg *At*(*Spare*, *Axle*) \wedge \neg *At*(*Spare*, *Trunk*)

\wedge \neg *At*(*Flat*, *Ground*) \wedge \neg *At*(*Flat*, *Axle*) \wedge \neg *At*(*Flat*, *Trunk*))

Diagram of the blocks-world problem



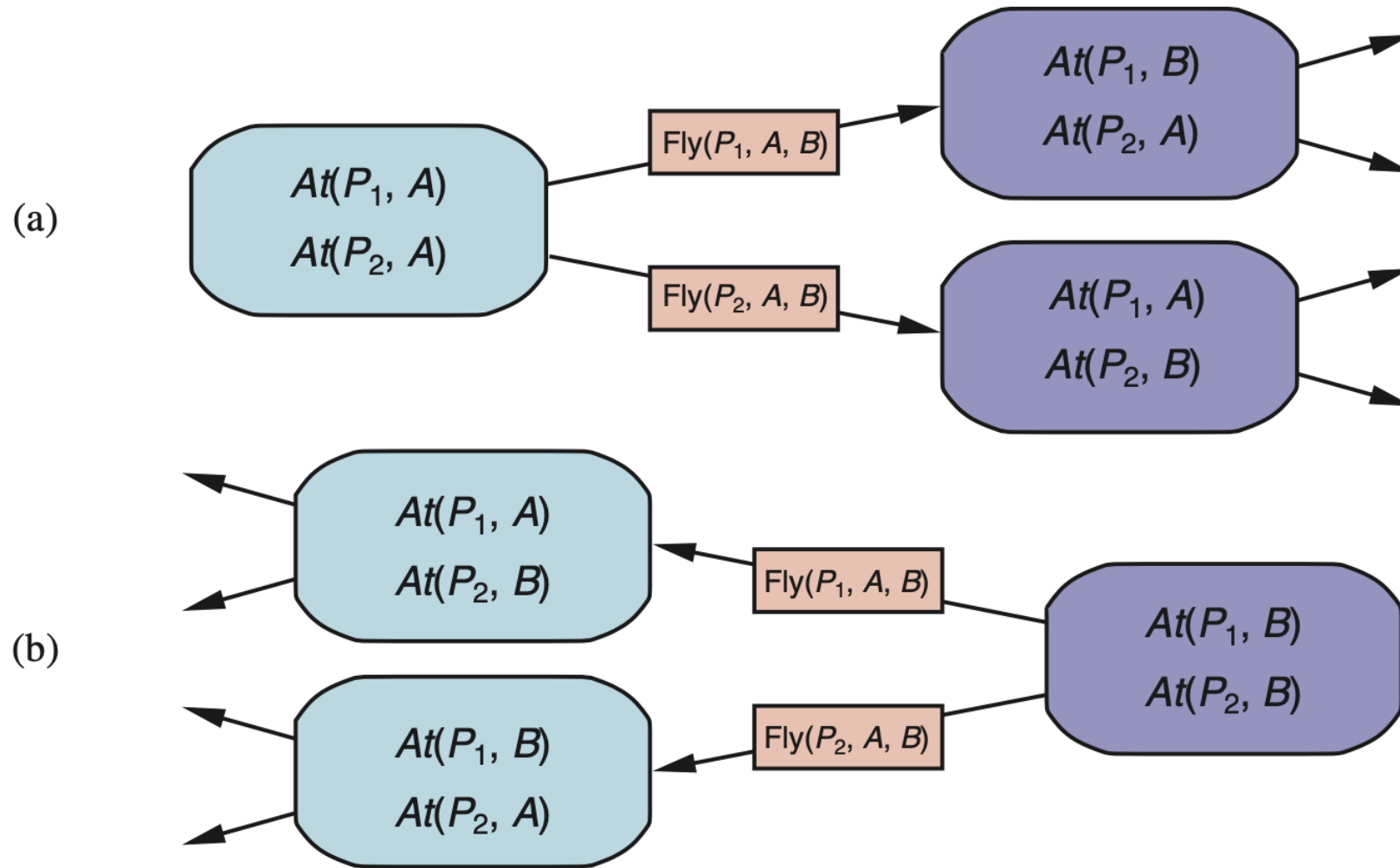
A planning problem in the blocks world: building a three-block tower

Init($On(A, Table) \wedge On(B, Table) \wedge On(C, A)$
 $\wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C) \wedge Clear(Table)$)
Goal($On(A, B) \wedge On(B, C)$)
Action(*Move*(b, x, y),
PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge$
 $(b \neq x) \wedge (b \neq y) \wedge (x \neq y)$,
EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)$)
Action(*MoveToTable*(b, x),
PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge Block(x)$,
EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x)$)

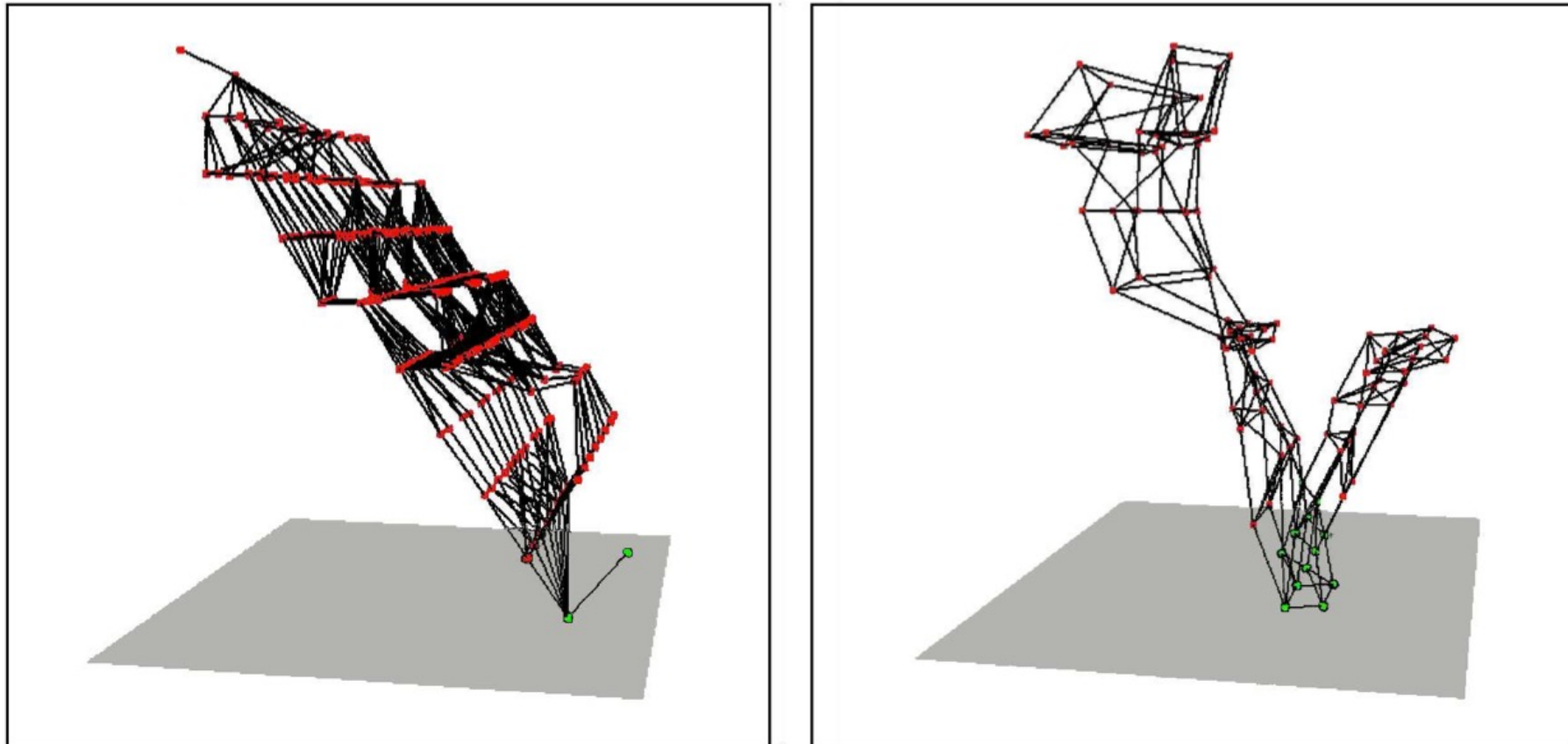
Two approaches to searching for a plan (a)

Forward (progression) search

(b) Backward (regression) search



Two state spaces from planning problems with the ignore-delete-lists heuristic



Definitions of possible refinements for two high-level actions

Refinement(*Go*(*Home*, *SFO*),
 STEPS: [*Drive*(*Home*, *SFO**LongTermParking*),
 Shuttle(*SFO**LongTermParking*, *SFO*)])

Refinement(*Go*(*Home*, *SFO*),
 STEPS: [*Taxi*(*Home*, *SFO*)])

Refinement(*Navigate*($[a, b]$, $[x, y]$),
 PRECOND: $a = x \wedge b = y$
 STEPS: [])

Refinement(*Navigate*($[a, b]$, $[x, y]$),
 PRECOND: *Connected*($[a, b]$, $[a - 1, b]$)
 STEPS: [*Left*, *Navigate*($[a - 1, b]$, $[x, y]$)])

Refinement(*Navigate*($[a, b]$, $[x, y]$),
 PRECOND: *Connected*($[a, b]$, $[a + 1, b]$)
 STEPS: [*Right*, *Navigate*($[a + 1, b]$, $[x, y]$)])

A breadth-first implementation of hierarchical forward planning search

function HIERARCHICAL-SEARCH(*problem*, *hierarchy*) **returns** a solution or *failure*

frontier \leftarrow a FIFO queue with [*Act*] as the only element

while *true* **do**

if IS-EMPTY(*frontier*) **then return** *failure*

plan \leftarrow POP(*frontier*) // chooses the shallowest plan in frontier

hla \leftarrow the first HLA in *plan*, or *null* if none

prefix, *suffix* \leftarrow the action subsequences before and after *hla* in *plan*

outcome \leftarrow RESULT(*problem*.INITIAL, *prefix*)

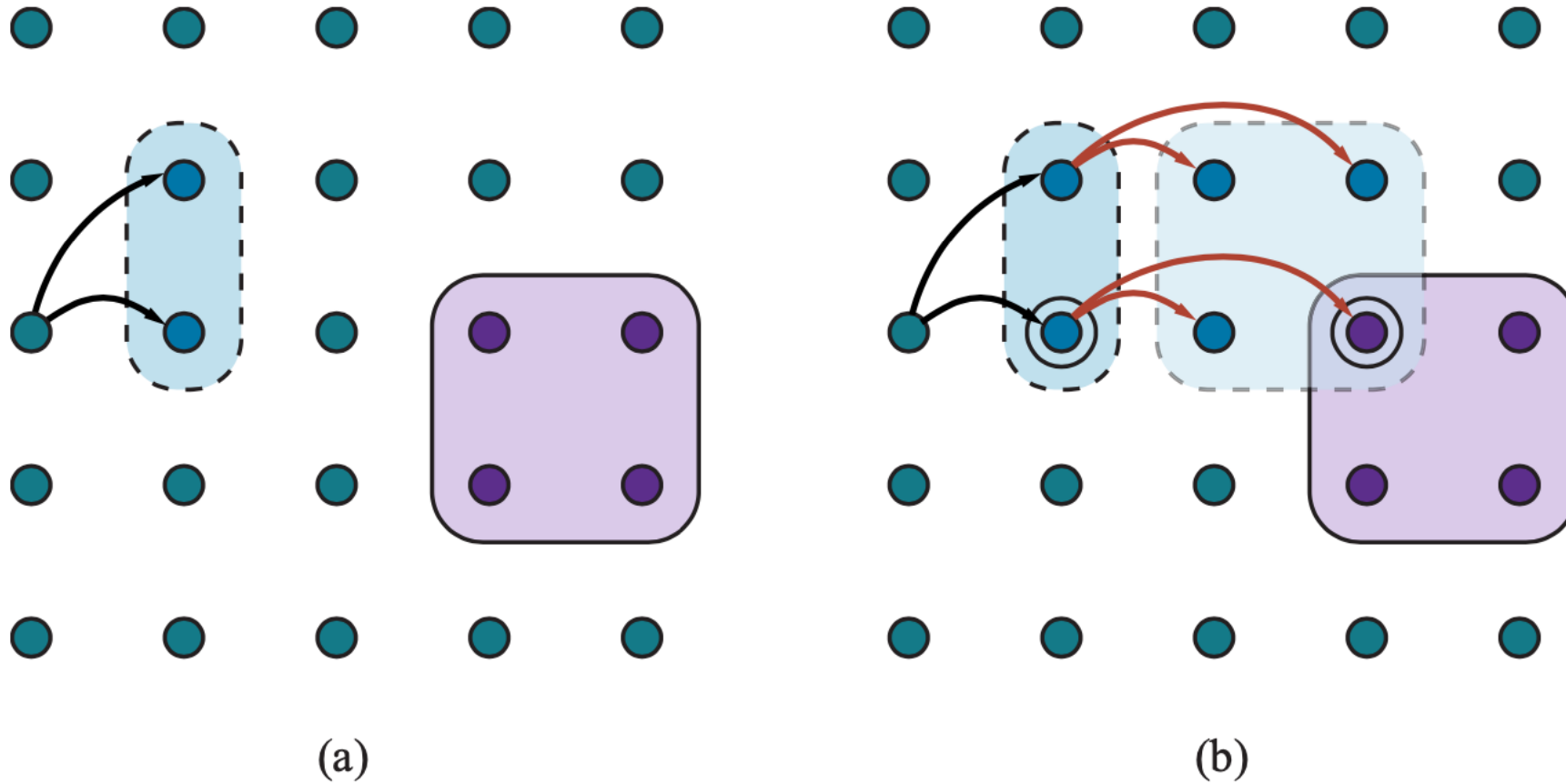
if *hla* is *null* **then** // so *plan* is primitive and *outcome* is its result

if *problem*.IS-GOAL(*outcome*) **then return** *plan*

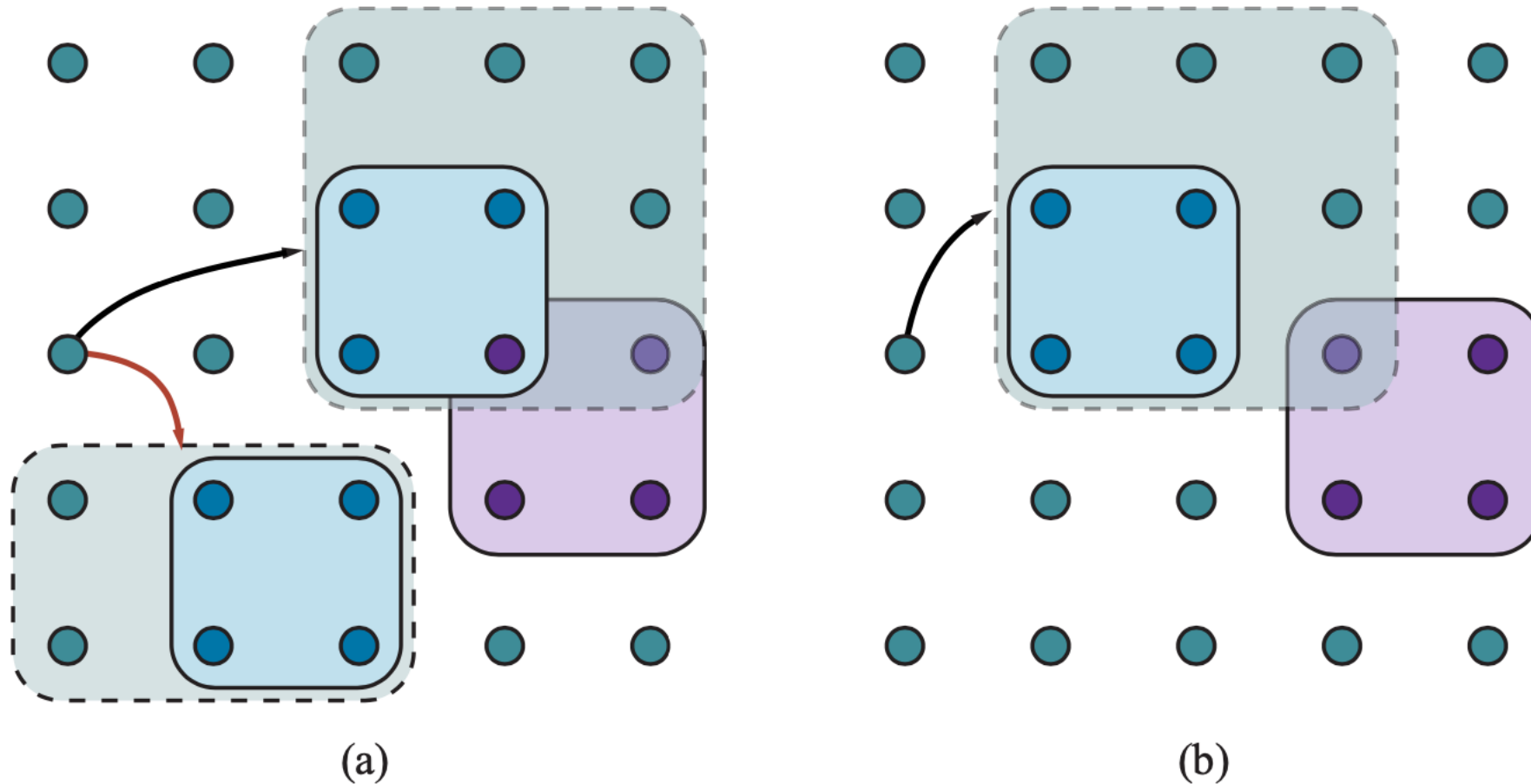
else for each *sequence* **in** REFINEMENTS(*hla*, *outcome*, *hierarchy*) **do**

 add APPEND(*prefix*, *sequence*, *suffix*) to *frontier*

Schematic examples of reachable sets



Goal achievement for high-level plans with approximate descriptions



A hierarchical planning algorithm

function ANGELIC-SEARCH(*problem*, *hierarchy*, *initialPlan*) **returns** solution or *fail*

frontier \leftarrow a FIFO queue with *initialPlan* as the only element

while *true* **do**

if EMPTY?(*frontier*) **then return** *fail*

plan \leftarrow POP(*frontier*) // chooses the shallowest node in *frontier*

if REACH⁺(*problem*.INITIAL, *plan*) intersects *problem*.GOAL **then**

if *plan* is primitive **then return** *plan* // REACH⁺ is exact for primitive plans

guaranteed \leftarrow REACH⁻(*problem*.INITIAL, *plan*) \cap *problem*.GOAL

if *guaranteed* \neq { } and MAKING-PROGRESS(*plan*, *initialPlan*) **then**

finalState \leftarrow any element of *guaranteed*

return DECOMPOSE(*hierarchy*, *problem*.INITIAL, *plan*, *finalState*)

hla \leftarrow some HLA in *plan*

prefix, *suffix* \leftarrow the action subsequences before and after *hla* in *plan*

outcome \leftarrow RESULT(*problem*.INITIAL, *prefix*)

for each *sequence* **in** REFINEMENTS(*hla*, *outcome*, *hierarchy*) **do**

frontier \leftarrow Insert(APPEND(*prefix*, *sequence*, *suffix*), *frontier*)

A hierarchical planning algorithm

Decompose solution

function DECOMPOSE(*hierarchy*, s_0 , *plan*, s_f) **returns** a solution

solution \leftarrow an empty plan

while *plan* is not empty **do**

action \leftarrow REMOVE-LAST(*plan*)

s_i \leftarrow a state in $\text{REACH}^-(s_0, \text{plan})$ such that $s_f \in \text{REACH}^-(s_i, \text{action})$

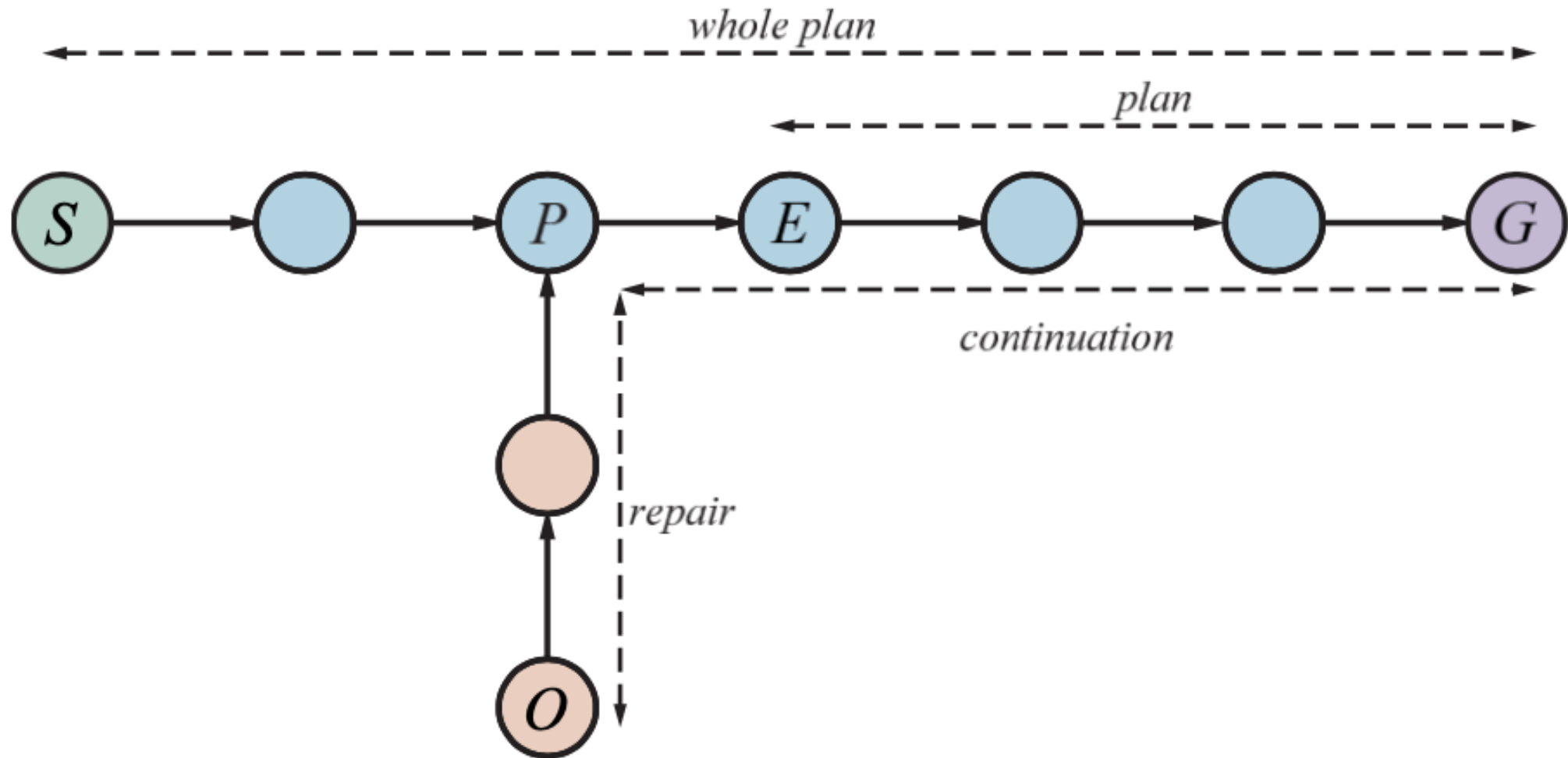
problem \leftarrow a problem with INITIAL = s_i and GOAL = s_f

solution \leftarrow APPEND(ANGELIC-SEARCH(*problem*, *hierarchy*, *action*), *solution*)

s_f \leftarrow s_i

return *solution*

At first, the sequence “whole plan” is expected to get the agent from S to G



A job-shop scheduling problem for assembling two cars, with resource constraints

Jobs({*AddEngine1* \prec *AddWheels1* \prec *Inspect1* },
{*AddEngine2* \prec *AddWheels2* \prec *Inspect2* })

Resources(*EngineHoists*(1), *WheelStations*(1), *Inspectors*(e2), *LugNuts*(500))

Action(*AddEngine1*, DURATION:30,
USE:*EngineHoists*(1))

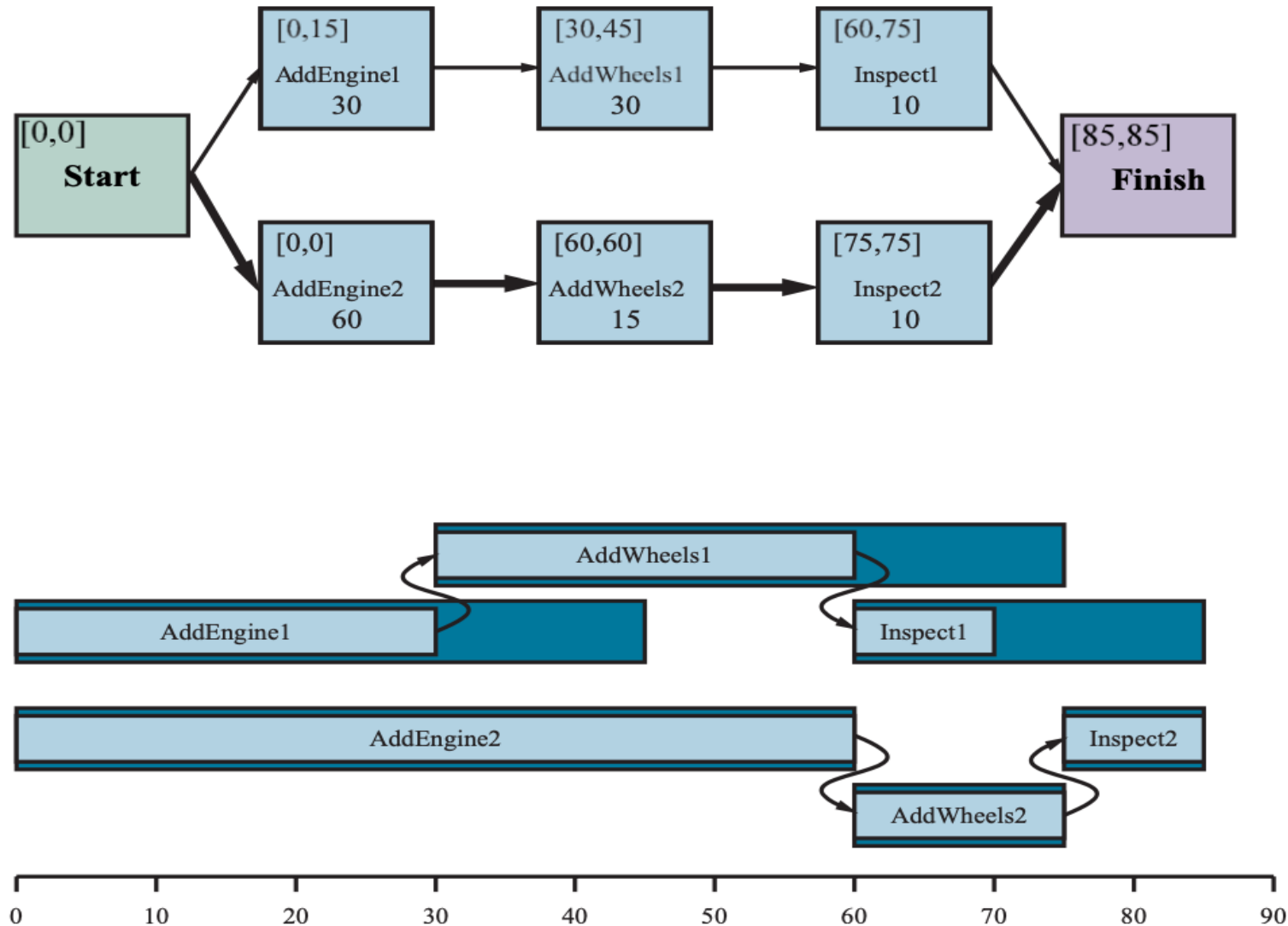
Action(*AddEngine2*, DURATION:60,
USE:*EngineHoists*(1))

Action(*AddWheels1*, DURATION:30,
CONSUME:*LugNuts*(20), USE:*WheelStations*(1))

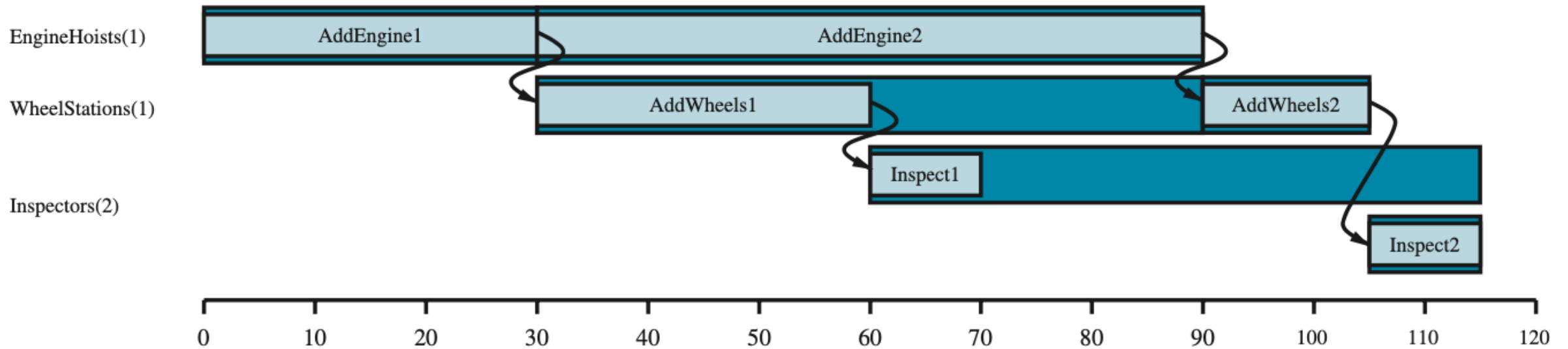
Action(*AddWheels2*, DURATION:15,
CONSUME:*LugNuts*(20), USE:*WheelStations*(1))

Action(*Inspect_i*, DURATION:10,
USE:*Inspectors*(1))

A representation of the temporal constraints for the job-shop scheduling problem



A solution to the job-shop scheduling problem



Artificial Intelligence: Uncertain Knowledge and Reasoning

Artificial Intelligence:

4. Uncertain Knowledge and Reasoning

- **Quantifying Uncertainty**
- **Probabilistic Reasoning**
- **Probabilistic Reasoning over Time**
- **Probabilistic Programming**
- **Making Simple Decisions**
- **Making Complex Decisions**
- **Multiagent Decision Making**

Quantifying Uncertainty

DT-Agent

A Decision-Theoretic Agent that Selects Rational Actions

function DT-AGENT(*percept*) **returns** an *action*

persistent: *belief_state*, probabilistic beliefs about the current state of the world
action, the agent's action

update *belief_state* based on *action* and *percept*

calculate outcome probabilities for actions,

 given action descriptions and current *belief_state*

select *action* with highest expected utility

 given probabilities of outcomes and utility information

return *action*

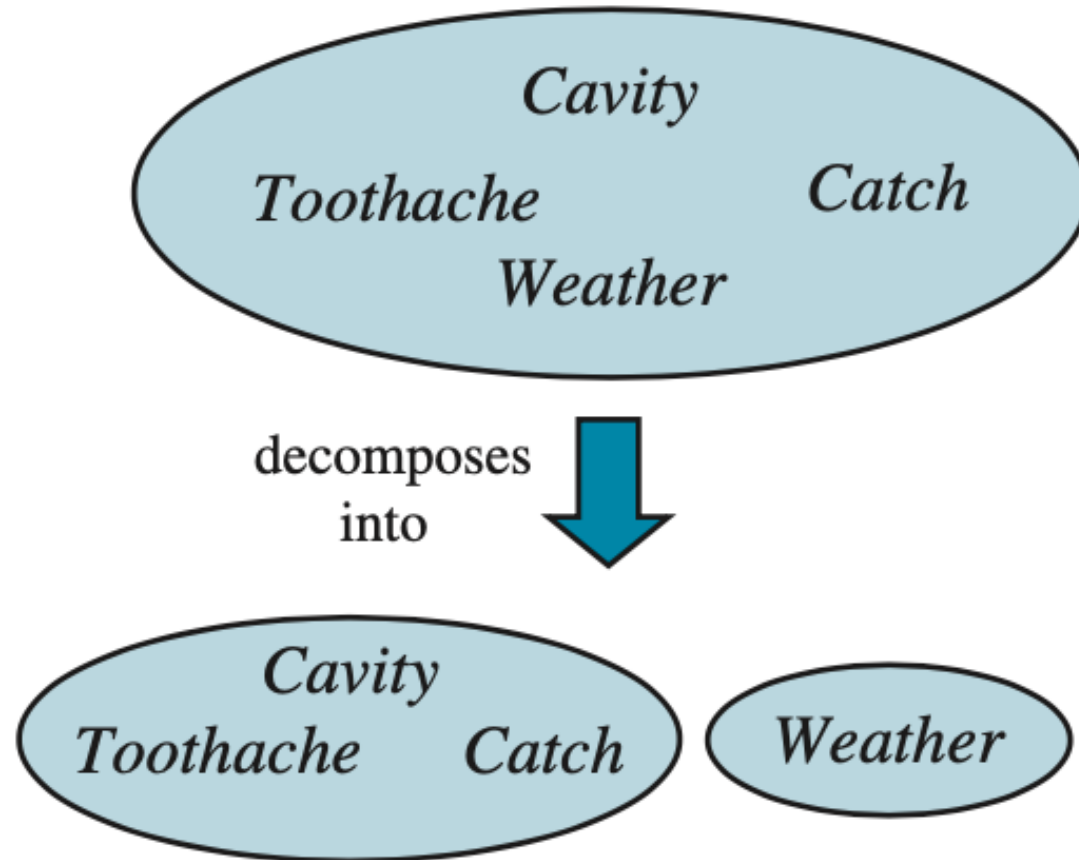
Agent 1 has inconsistent beliefs

Proposition	Agent 1's belief	Agent 2 bets	Agent 1 bets	Agent 1 payoffs for each outcome			
				a, b	$a, \neg b$	$\neg a, b$	$\neg a, \neg b$
a	0.4	\$4 on a	\$6 on $\neg a$	-\$6	-\$6	\$4	\$4
b	0.3	\$3 on b	\$7 on $\neg b$	-\$7	\$3	-\$7	\$3
$a \vee b$	0.8	\$2 on $\neg(a \vee b)$	\$8 on $a \vee b$	\$2	\$2	\$2	-\$8
				-\$11	-\$1	-\$1	-\$1

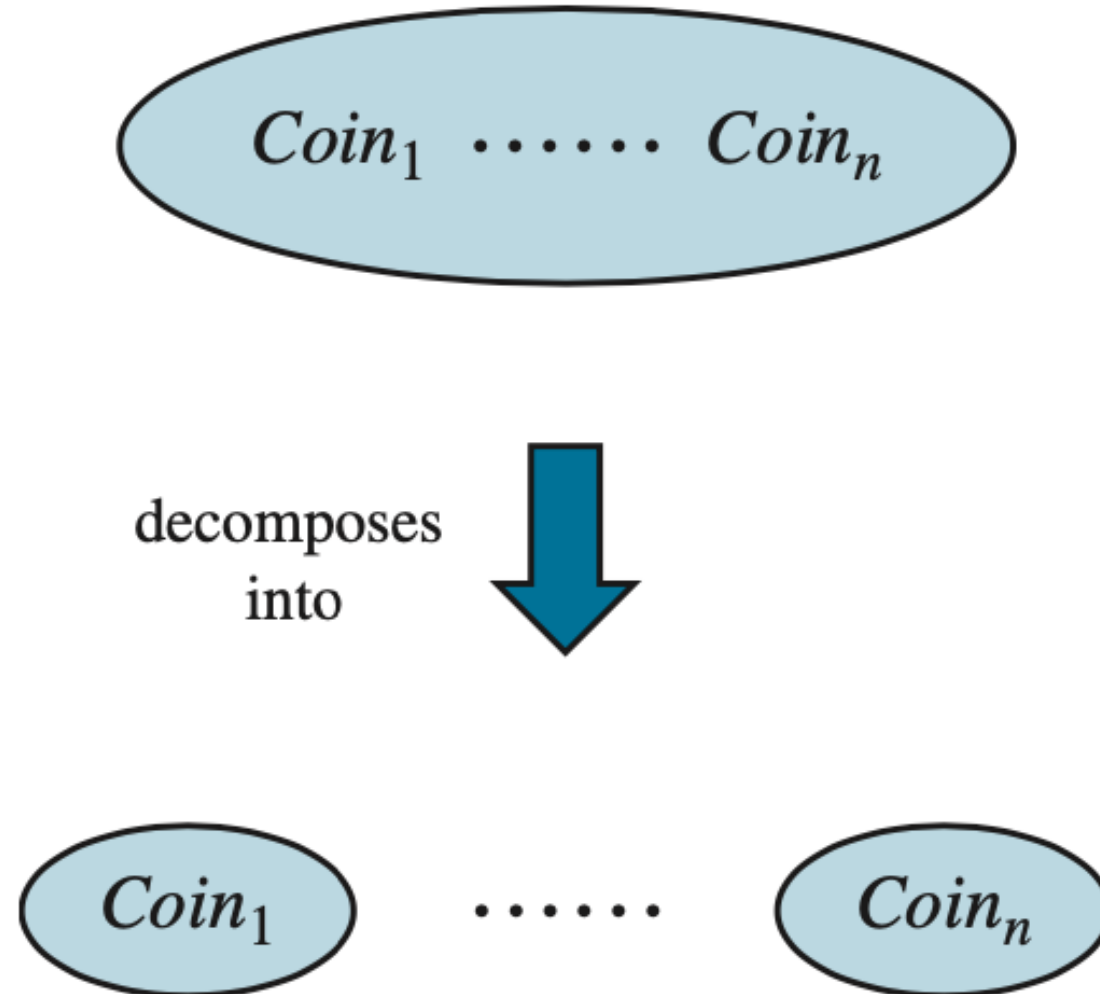
A full joint distribution for the Toothache, Cavity, Catch world

	<i>toothache</i>		\neg <i>toothache</i>	
	<i>catch</i>	\neg <i>catch</i>	<i>catch</i>	\neg <i>catch</i>
<i>cavity</i>	0.108	0.012	0.072	0.008
\neg <i>cavity</i>	0.016	0.064	0.144	0.576

Weather and Dental problems are independent



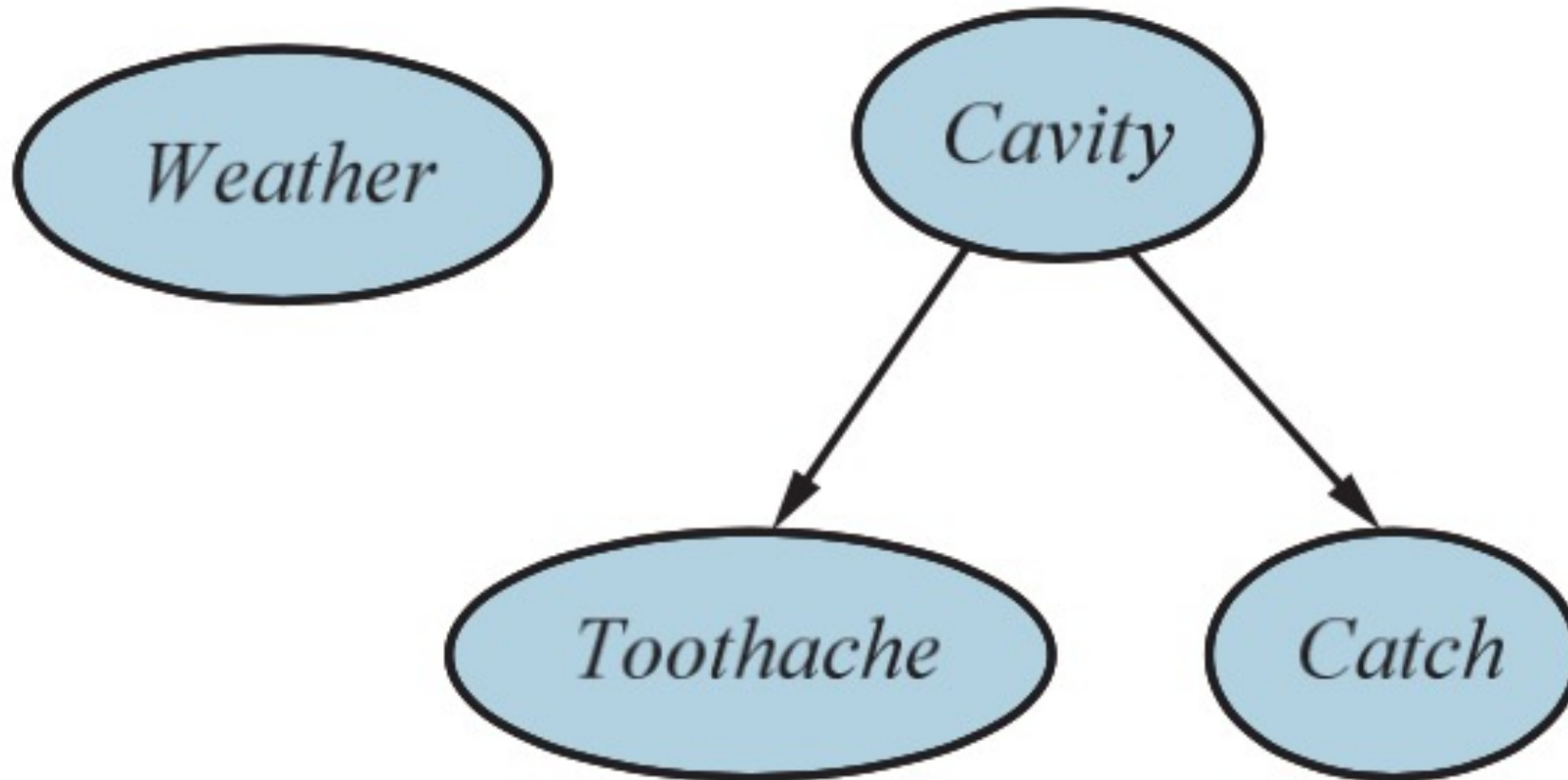
Coin flips are independent



Probabilistic Reasoning

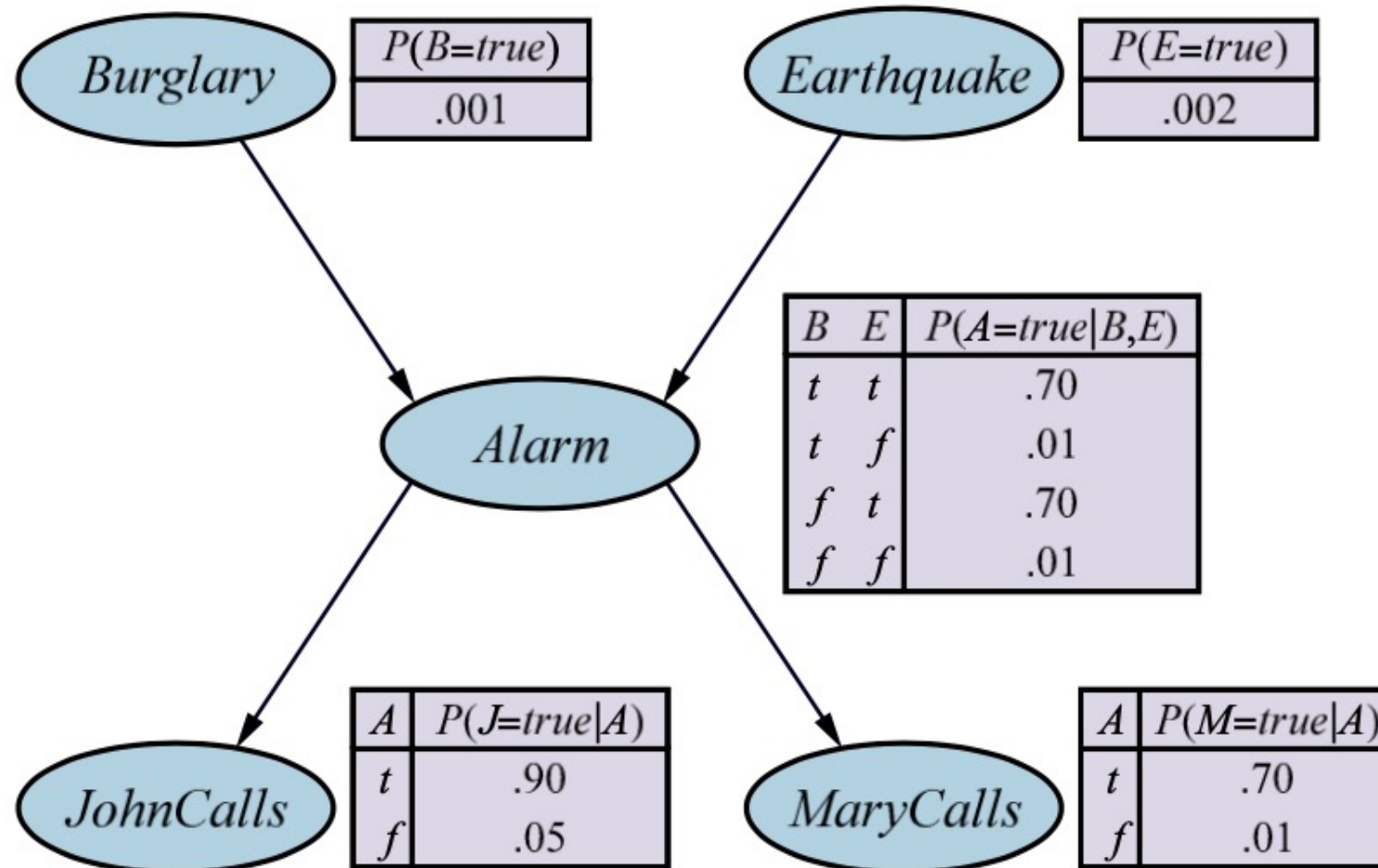
A Simple Bayesian Network

Weather is independent to the other three variables.
Toothache and Catch are conditionally independent, given Cavity.



A Typical Bayesian Network

Topology and the Conditional Probability Tables (CPTs)



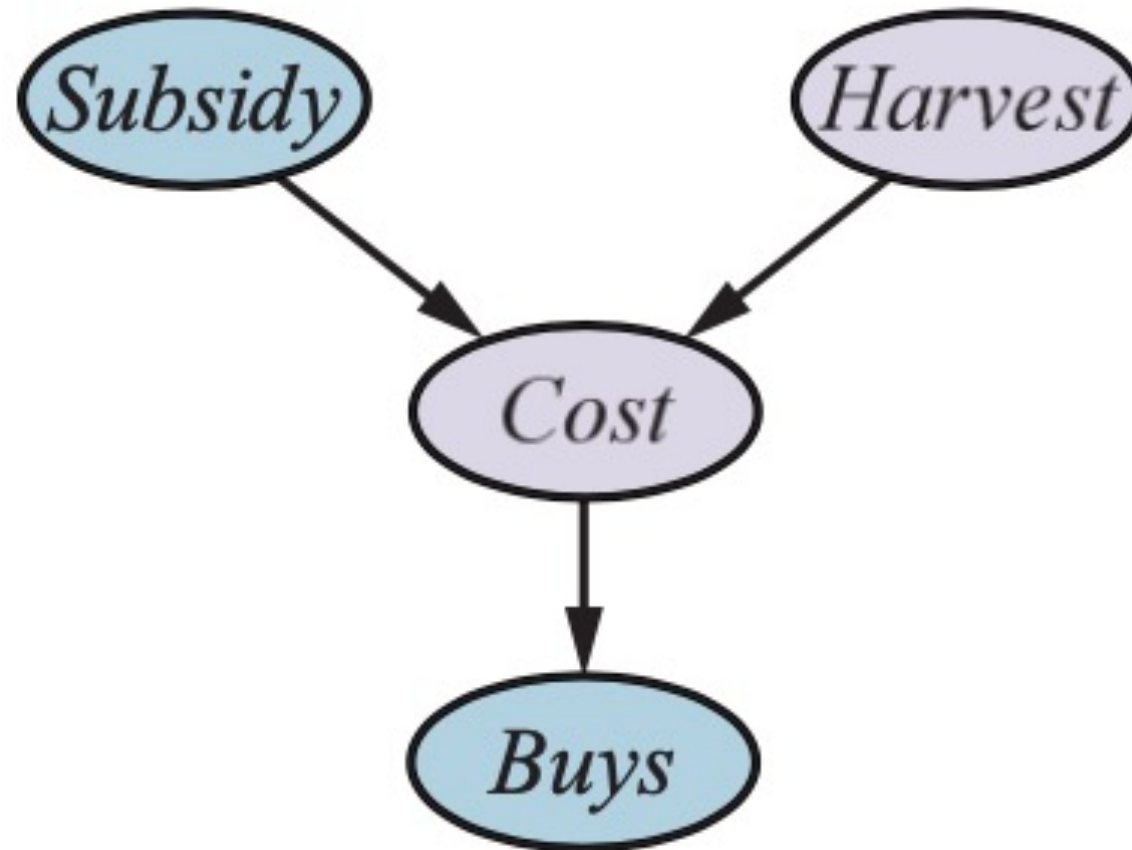
Conditional Probability Table

for $P(\text{Fever} \mid \text{Cold, Flu, Malaria})$

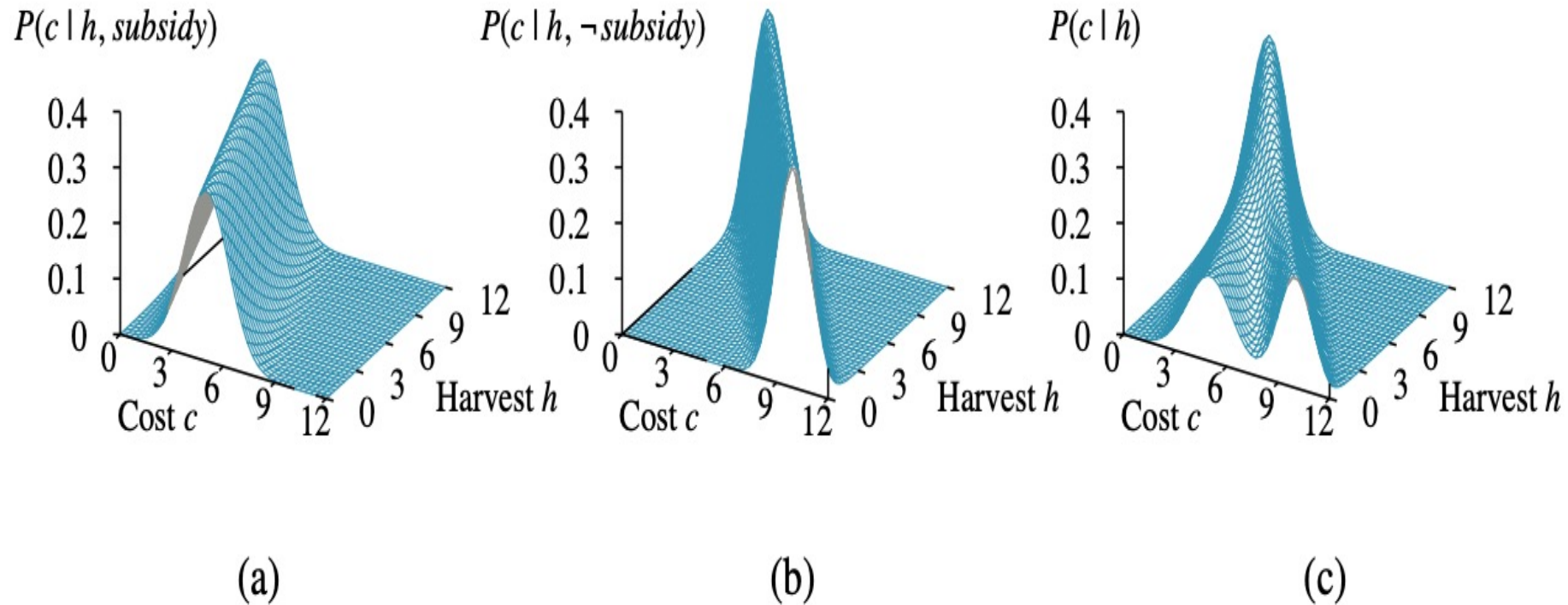
<i>Cold</i>	<i>Flu</i>	<i>Malaria</i>	$P(\text{fever} \mid \cdot)$	$P(\neg\text{fever} \mid \cdot)$
<i>f</i>	<i>f</i>	<i>f</i>	0.0	1.0
<i>f</i>	<i>f</i>	<i>t</i>	0.9	0.1
<i>f</i>	<i>t</i>	<i>f</i>	0.8	0.2
<i>f</i>	<i>t</i>	<i>t</i>	0.98	$0.02 = 0.2 \times 0.1$
<i>t</i>	<i>f</i>	<i>f</i>	0.4	0.6
<i>t</i>	<i>f</i>	<i>t</i>	0.94	$0.06 = 0.6 \times 0.1$
<i>t</i>	<i>t</i>	<i>f</i>	0.88	$0.12 = 0.6 \times 0.2$
<i>t</i>	<i>t</i>	<i>t</i>	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

A Simple Network

with discrete variables (Subsidy and Buys)
and continuous variables (Harvest and Cost)

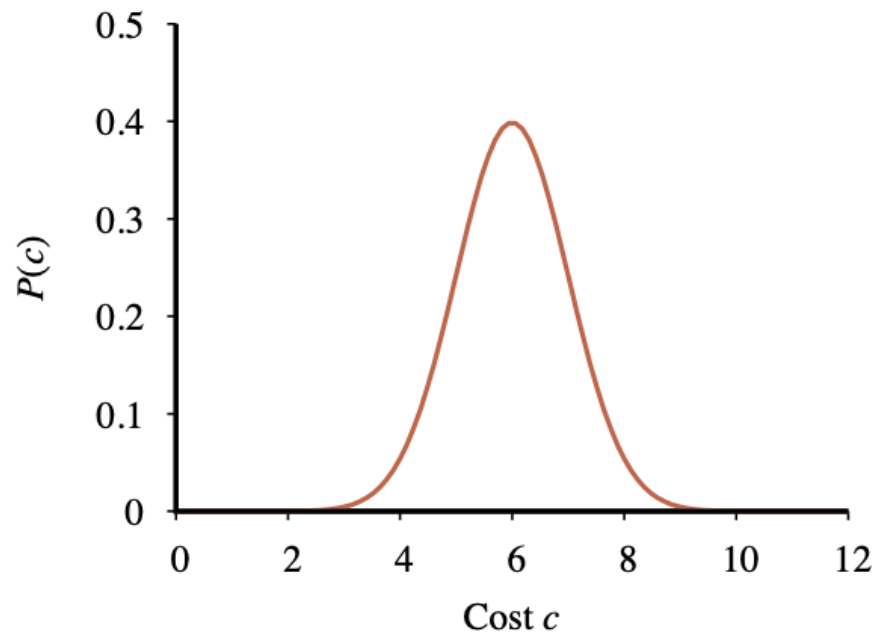


Probability distribution over Cost as a function of Harvest size

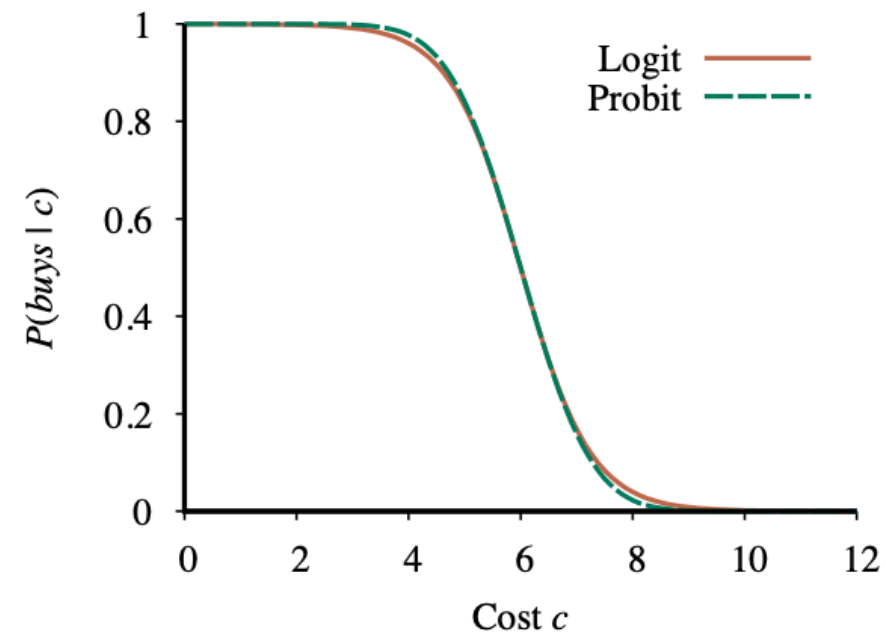


distribution $P(\text{Cost} | \text{Harvest})$,
obtained by summing over the
two subsidy cases.

A normal (Gaussian) distribution for the cost threshold



(a)

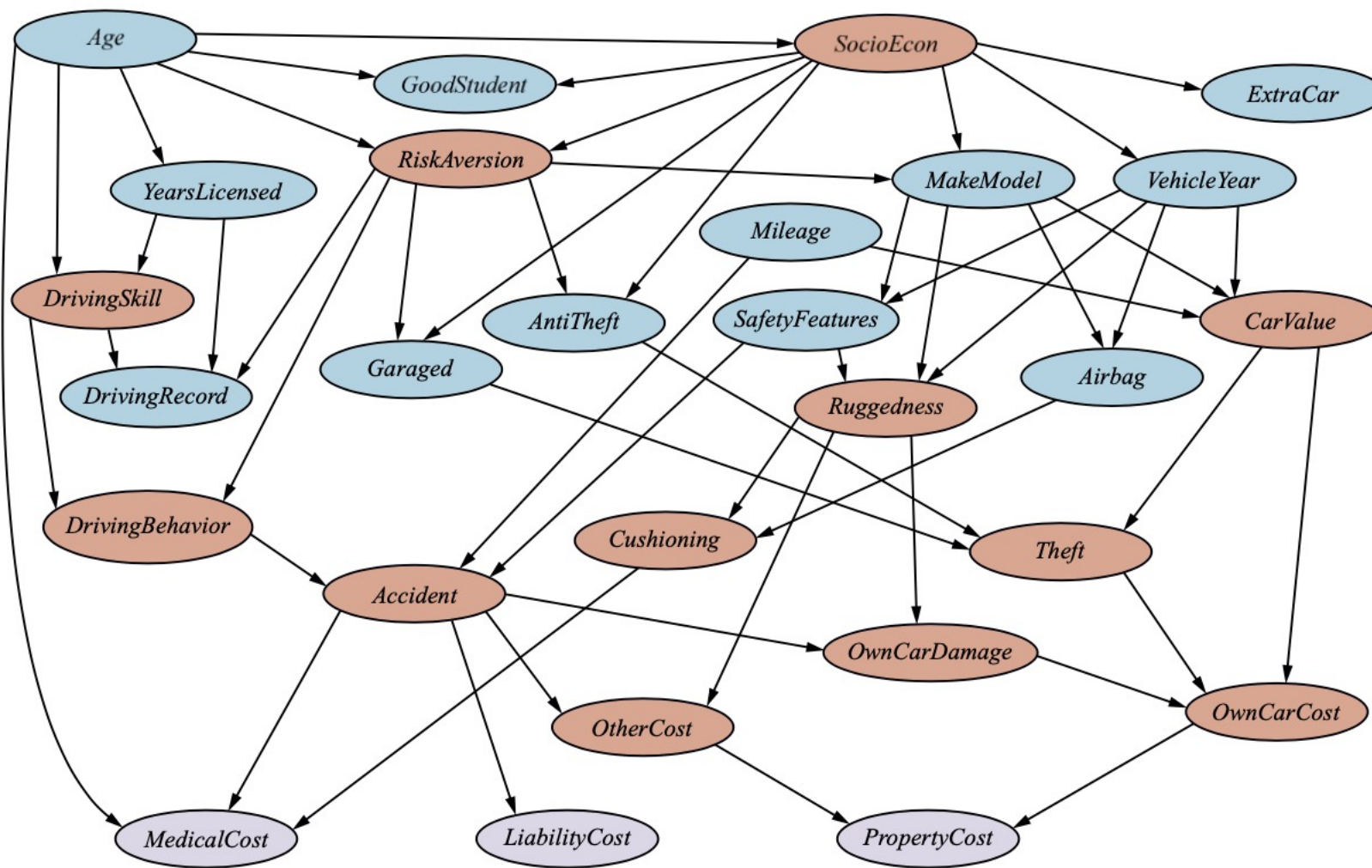


(b)

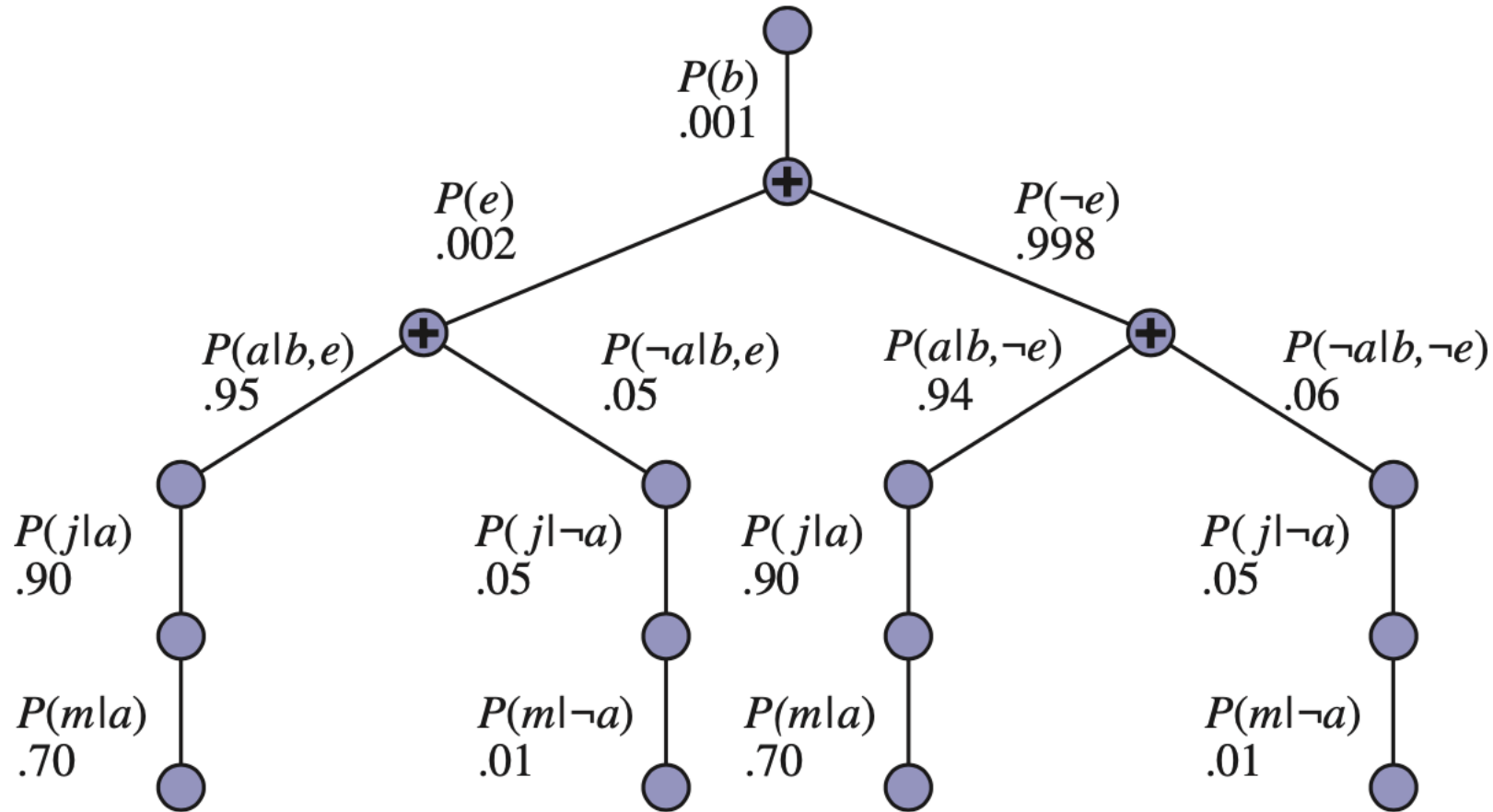
Expit and Probit models for the
probability of buys given cost

A Bayesian Network

for evaluating car insurance applications



The structure of the expression



The Enumeration Algorithm for Exact Inference in Bayes Nets

function ENUMERATION-ASK(X, \mathbf{e}, bn) **returns** a distribution over X

inputs: X , the query variable

\mathbf{e} , observed values for variables \mathbf{E}

bn , a Bayes net with variables $vars$

$\mathbf{Q}(X) \leftarrow$ a distribution over X , initially empty

for each value x_i of X **do**

$\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($vars, \mathbf{e}_{x_i}$)

where \mathbf{e}_{x_i} is \mathbf{e} extended with $X = x_i$

return NORMALIZE($\mathbf{Q}(X)$)

function ENUMERATE-ALL($vars, \mathbf{e}$) **returns** a real number

if EMPTY?($vars$) **then return** 1.0

$V \leftarrow$ FIRST($vars$)

if V is an evidence variable with value v in \mathbf{e}

then return $P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e})

else return $\sum_v P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e}_v)

where \mathbf{e}_v is \mathbf{e} extended with $V = v$

Pointwise Multiplication

$$f(X, Y) \times g(Y, Z) = h(X, Y, Z)$$

X	Y	$f(X, Y)$	Y	Z	$g(Y, Z)$	X	Y	Z	$h(X, Y, Z)$
t	t	.3	t	t	.2	t	t	t	$.3 \times .2 = .06$
t	f	.7	t	f	.8	t	t	f	$.3 \times .8 = .24$
f	t	.9	f	t	.6	t	f	t	$.7 \times .6 = .42$
f	f	.1	f	f	.4	t	f	f	$.7 \times .4 = .28$
						f	t	t	$.9 \times .2 = .18$
						f	t	f	$.9 \times .8 = .72$
						f	f	t	$.1 \times .6 = .06$
						f	f	f	$.1 \times .4 = .04$

The Variable Elimination Algorithm for Exact Inference in Bayes Nets

function ELIMINATION-ASK(X, \mathbf{e}, bn) **returns** a distribution over X

inputs: X , the query variable

\mathbf{e} , observed values for variables \mathbf{E}

bn , a Bayesian network with variables $vars$

$factors \leftarrow []$

for each V **in** ORDER($vars$) **do**

$factors \leftarrow [\text{MAKE-FACTOR}(V, \mathbf{e})] + factors$

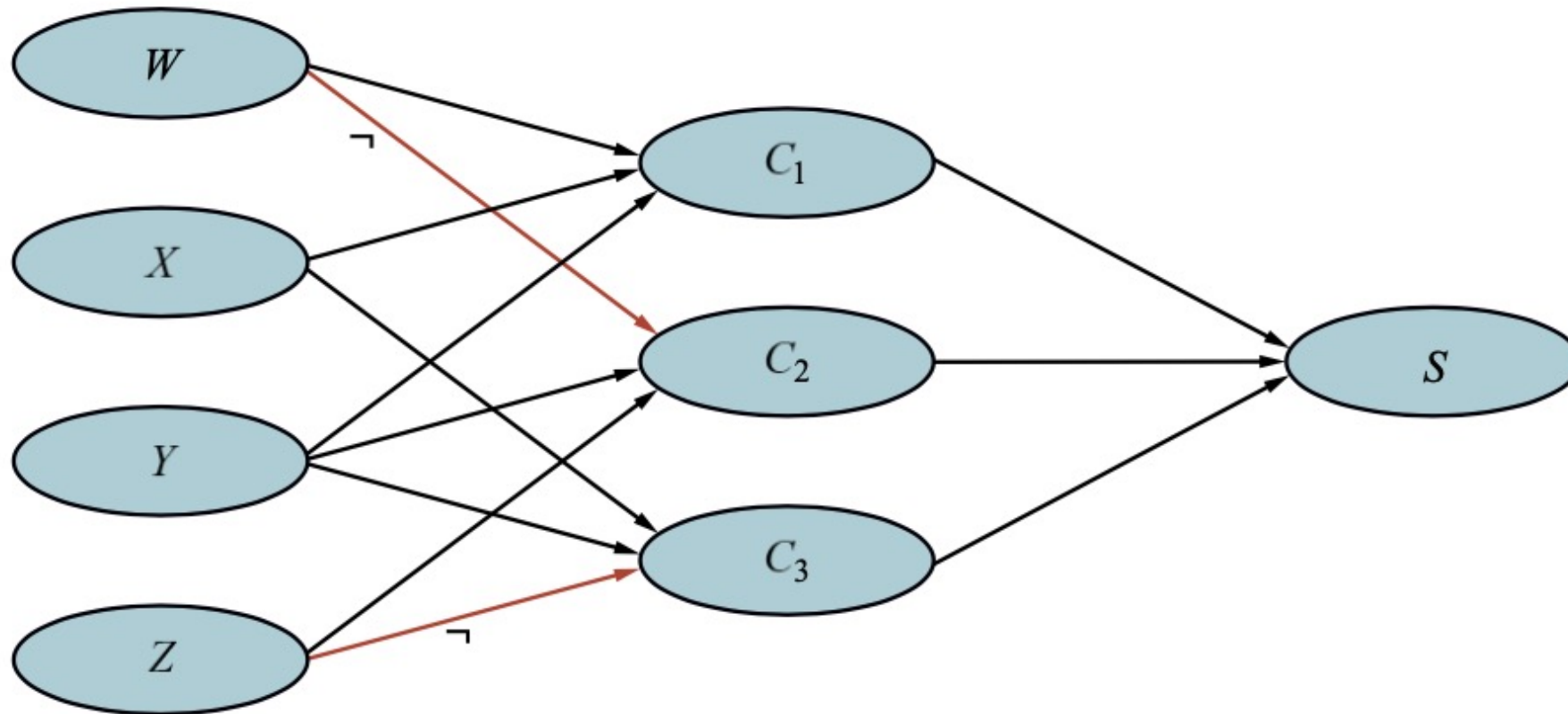
if V is a hidden variable **then** $factors \leftarrow \text{SUM-OUT}(V, factors)$

return NORMALIZE(PPOINTWISE-PRODUCT($factors$))

Bayes Net Encoding

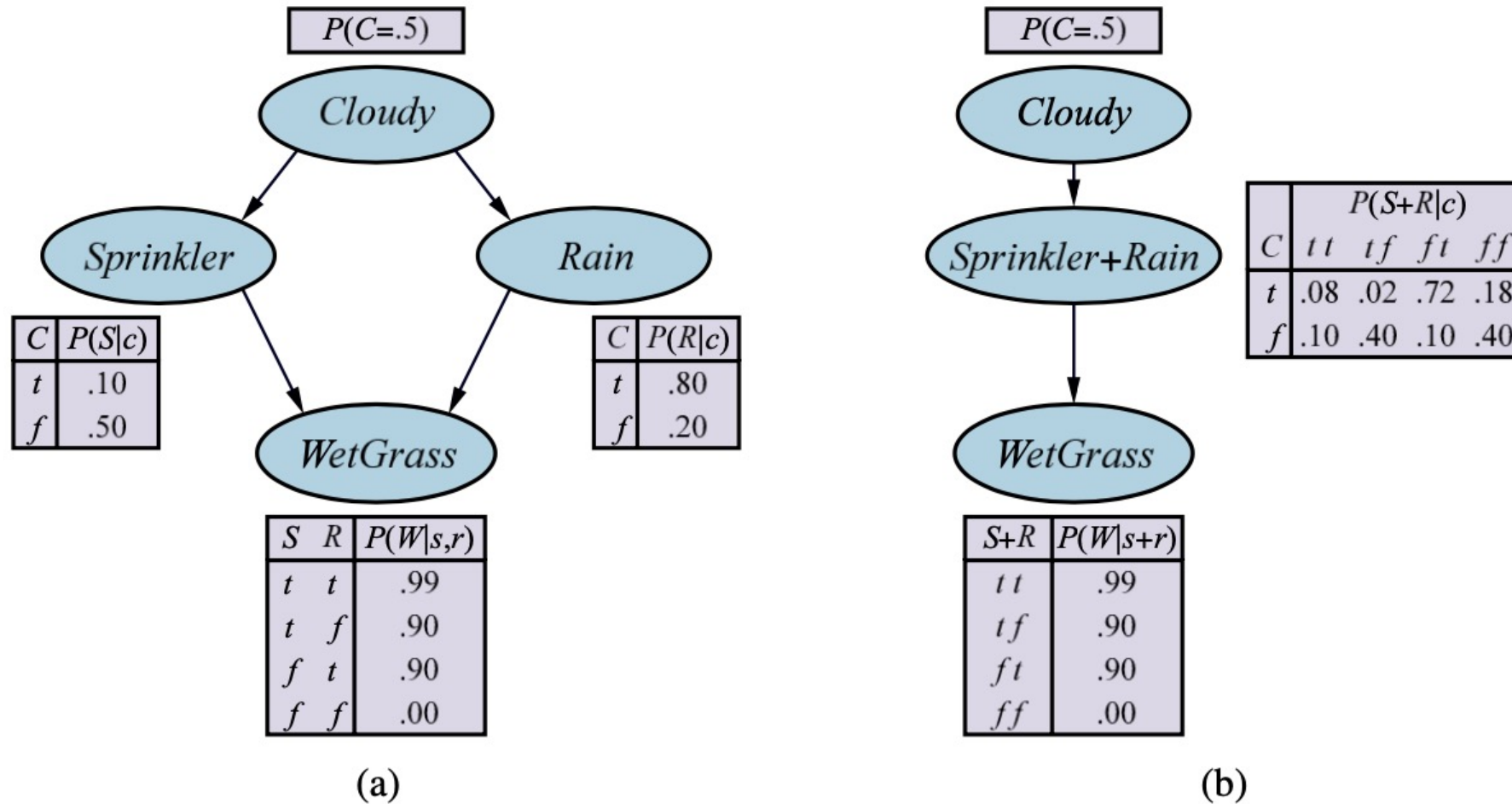
of the 3-CNF (Conjunctive Normal Form) Sentence

$$(W \vee X \vee Y) \wedge (\neg W \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z)$$



Multiply Connected Network

(b) A clustered equivalent



A Sampling Algorithm

that generates events from a Bayesian network

function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn

inputs: bn , a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \dots, X_n)$

$\mathbf{x} \leftarrow$ an event with n elements

for each variable X_i **in** X_1, \dots, X_n **do**

$\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid \text{parents}(X_i))$

return \mathbf{x}

The Rejection-Sampling Algorithm

for answering queries given evidence in a Bayesian network

function REJECTION-SAMPLING(X, \mathbf{e}, bn, N) **returns** an estimate of $\mathbf{P}(X | \mathbf{e})$

inputs: X , the query variable

\mathbf{e} , observed values for variables \mathbf{E}

bn , a Bayesian network

N , the total number of samples to be generated

local variables: \mathbf{C} , a vector of counts for each value of X , initially zero

for $j = 1$ **to** N **do**

$\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)

if \mathbf{x} is consistent with \mathbf{e} **then**

$\mathbf{C}[j] \leftarrow \mathbf{C}[j]+1$ where x_j is the value of X in \mathbf{x}

return NORMALIZE(\mathbf{C})

The Likelihood-Weighting Algorithm for inference in Bayesian networks

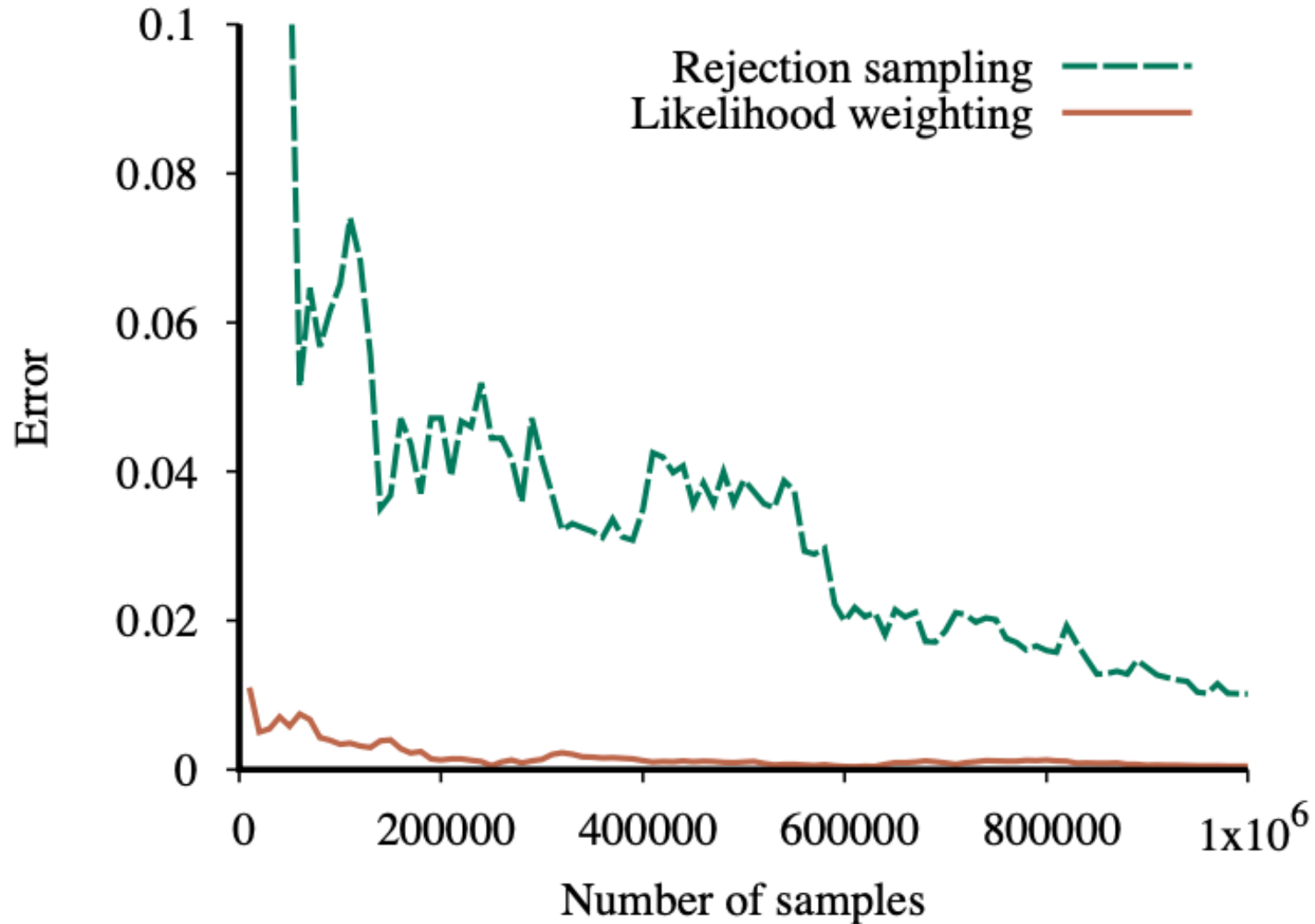
function LIKELIHOOD-WEIGHTING(X, \mathbf{e}, bn, N) **returns** an estimate of $\mathbf{P}(X | \mathbf{e})$
inputs: X , the query variable
 \mathbf{e} , observed values for variables \mathbf{E}
 bn , a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \dots, X_n)$
 N , the total number of samples to be generated
local variables: \mathbf{W} , a vector of weighted counts for each value of X , initially zero

for $j = 1$ **to** N **do**
 $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE(bn, \mathbf{e})
 $\mathbf{W}[j] \leftarrow \mathbf{W}[j] + w$ where x_j is the value of X in \mathbf{x}
return NORMALIZE(\mathbf{W})

function WEIGHTED-SAMPLE(bn, \mathbf{e}) **returns** an event and a weight

$w \leftarrow 1$; $\mathbf{x} \leftarrow$ an event with n elements, with values fixed from \mathbf{e}
for $i = 1$ **to** n **do**
 if X_i is an evidence variable with value x_{ij} in \mathbf{e}
 then $w \leftarrow w \times P(X_i = x_{ij} | \text{parents}(X_i))$
 else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i | \text{parents}(X_i))$
return \mathbf{x}, w

Performance of rejection sampling and likelihood weighting on the insurance network



The Gibbs Sampling Algorithm

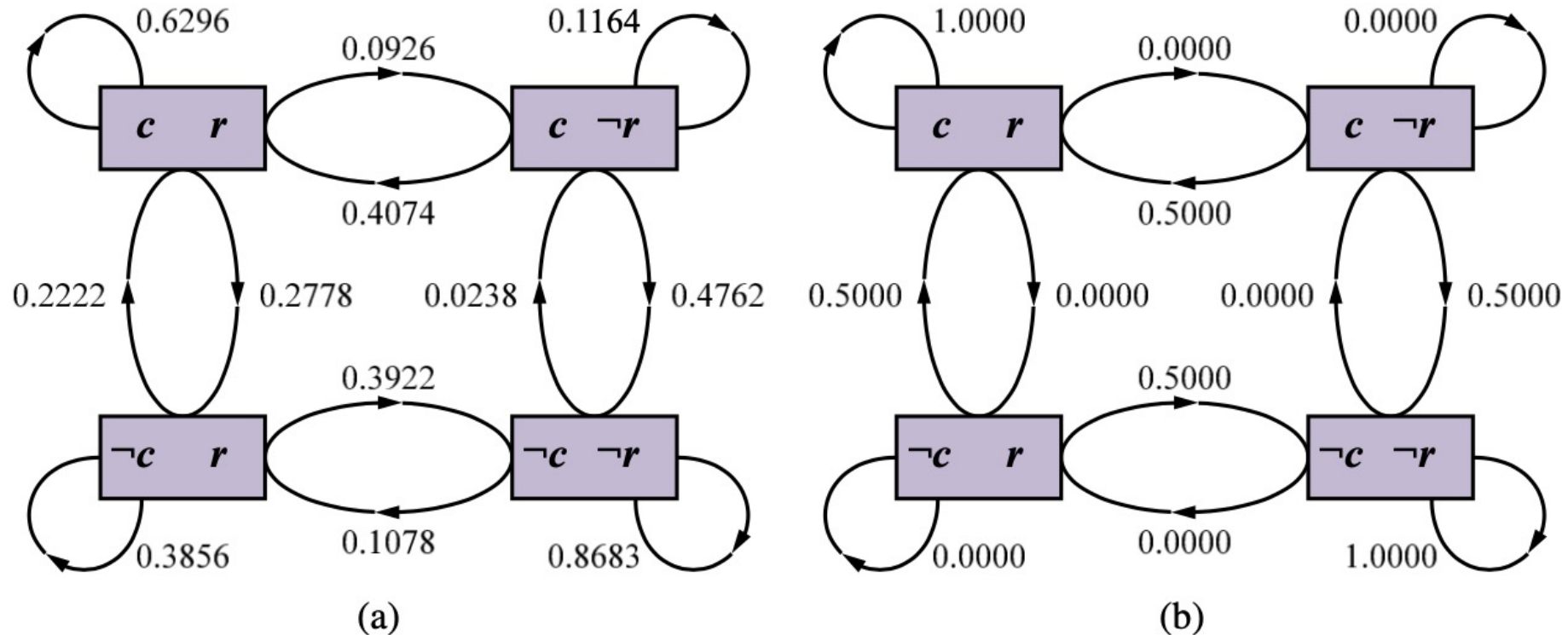
for approximate inference in Bayes nets

function GIBBS-ASK(X, \mathbf{e}, bn, N) **returns** an estimate of $\mathbf{P}(X | \mathbf{e})$
local variables: \mathbf{C} , a vector of counts for each value of X , initially zero
 \mathbf{Z} , the nonevidence variables in bn
 \mathbf{x} , the current state of the network, initialized from \mathbf{e}

initialize \mathbf{x} with random values for the variables in \mathbf{Z}
for $k = 1$ **to** N **do**
 choose any variable Z_i from \mathbf{Z} according to any distribution $\rho(i)$
 set the value of Z_i in \mathbf{x} by sampling from $\mathbf{P}(Z_i | mb(Z_i))$
 $\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$ where x_j is the value of X in \mathbf{x}
return NORMALIZE(\mathbf{C})

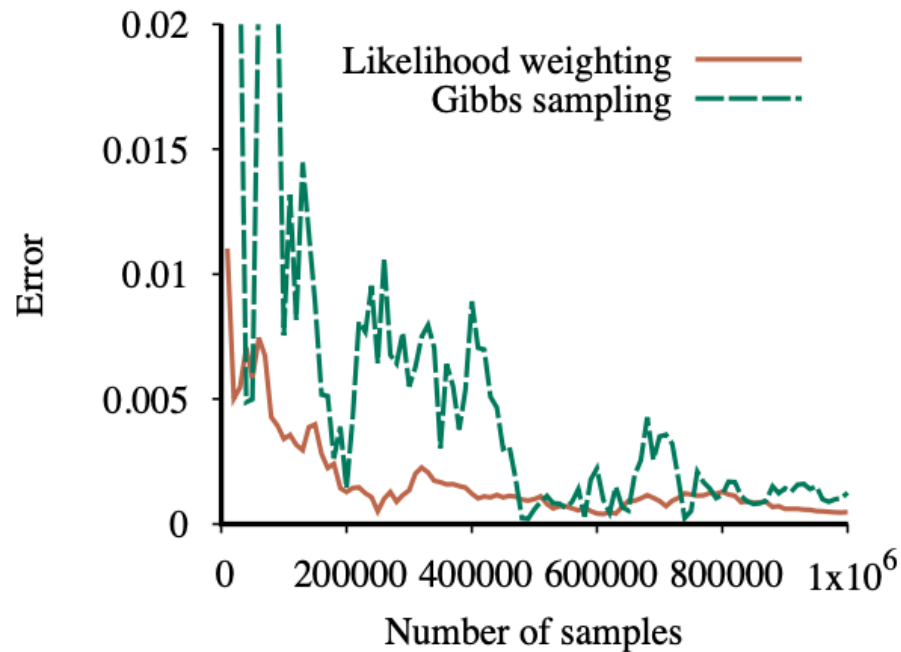
The States and Transition Probabilities of the Markov Chain

for the query $P(\text{Rain} \mid \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$



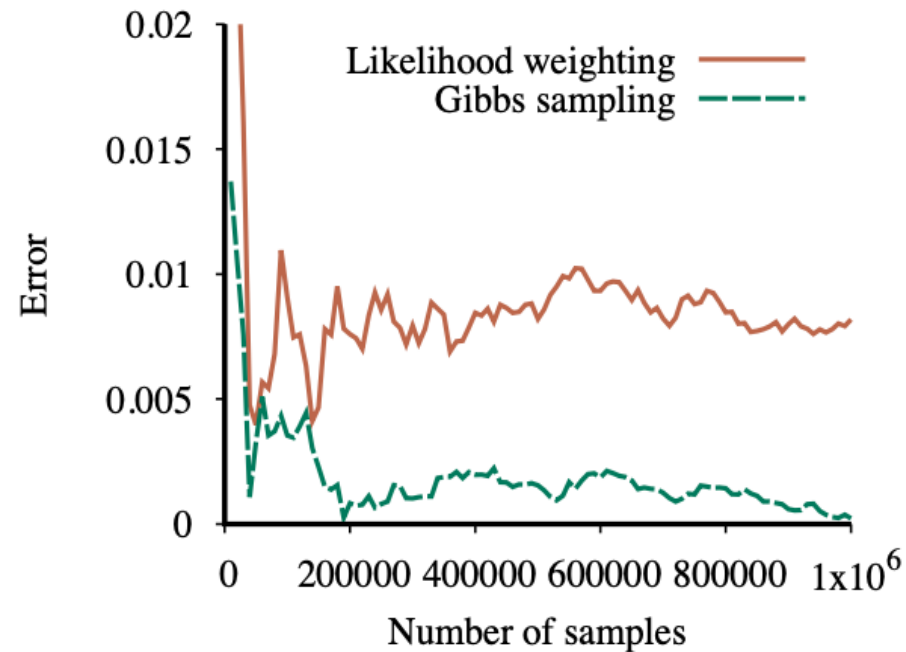
**Transition Probabilities
when the CPT for Rain constrains it
to have the same value as Cloudy**

Performance of Gibbs sampling compared to likelihood weighting on the car insurance network



(a)

for the standard query on PropertyCost

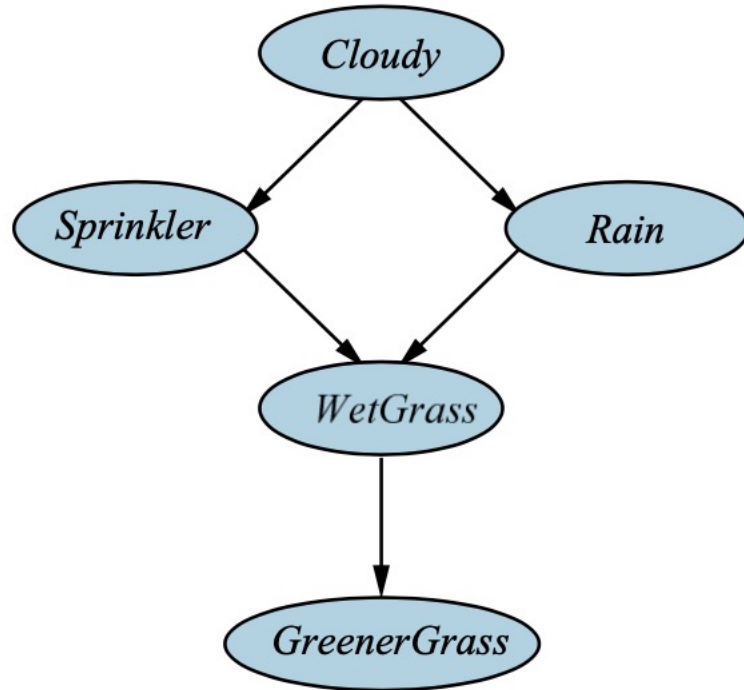


(b)

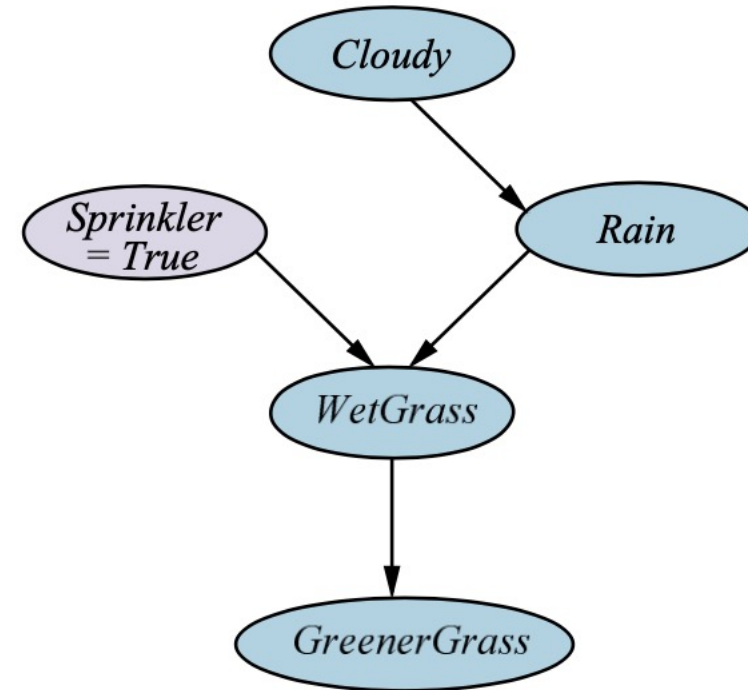
for the case where the output variables are observed and Age is the query variable

A Causal Bayesian Network

representing cause-effect relations among five variables



(a)



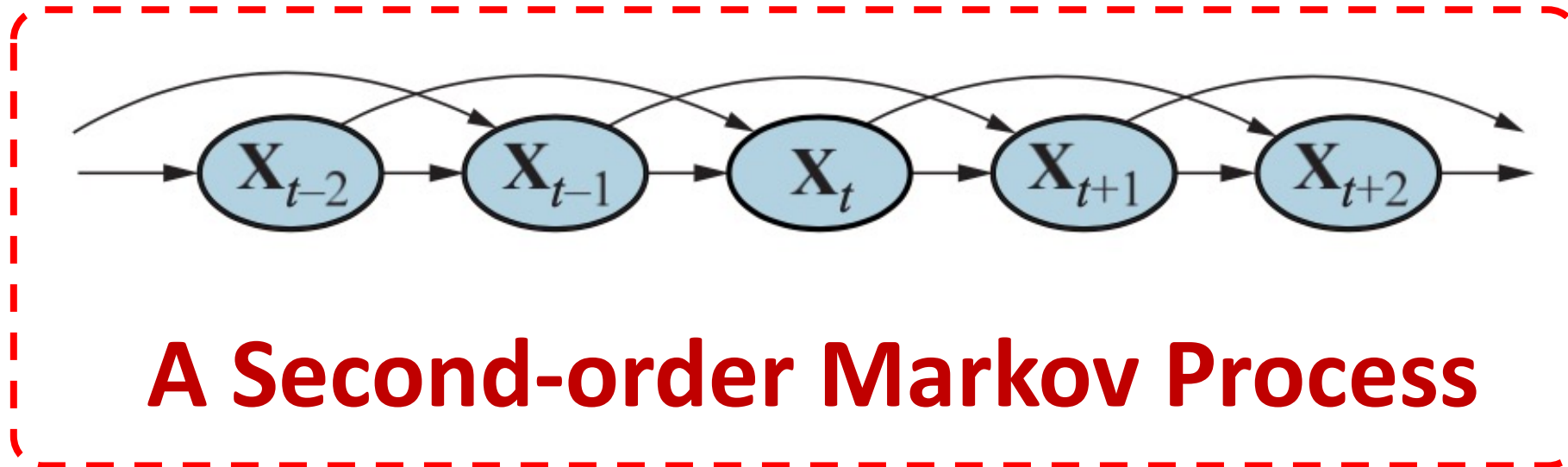
(b)

The network after performing the action
"turn Sprinkler on."

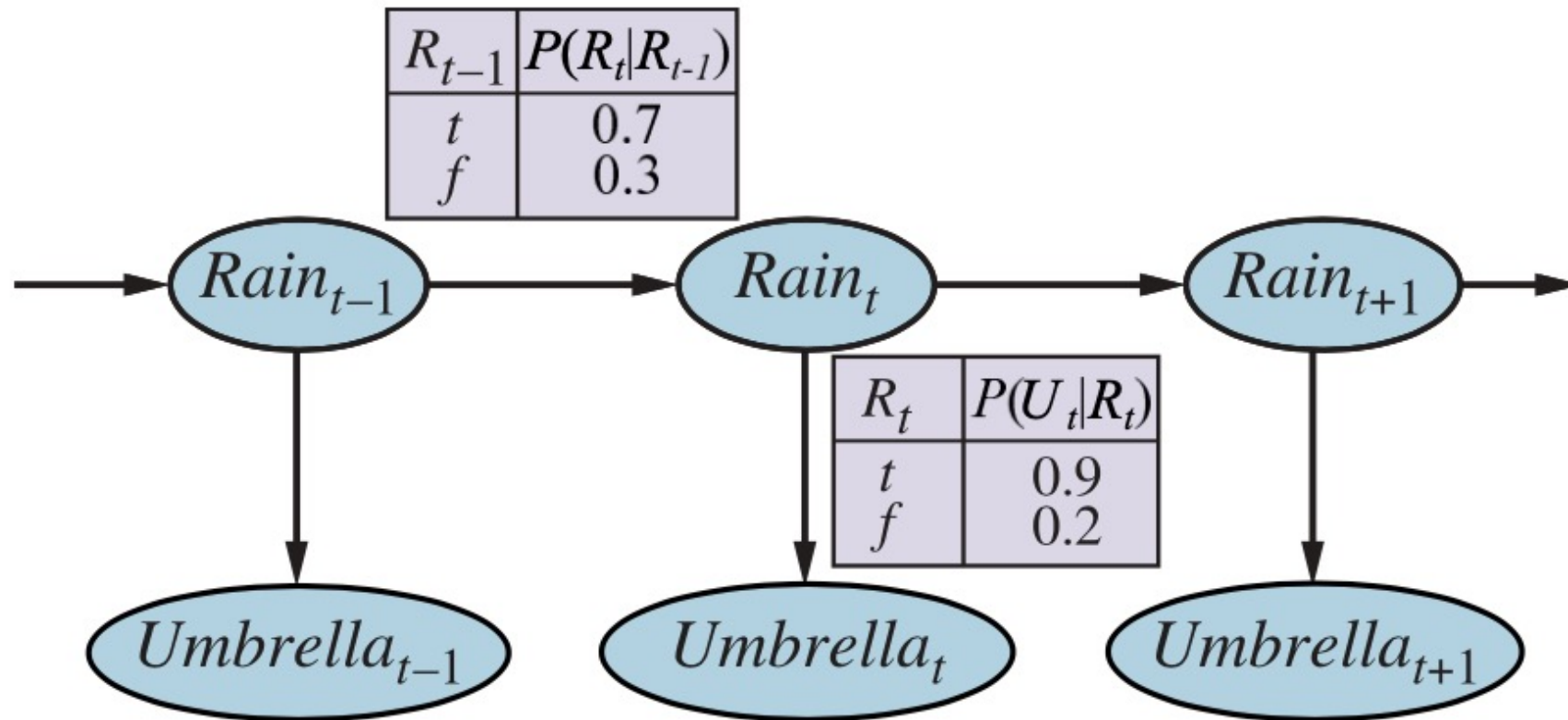
Probabilistic Reasoning over Time

Bayesian network structure

corresponding to a First-order Markov Process
with state defined by the variables X_t .

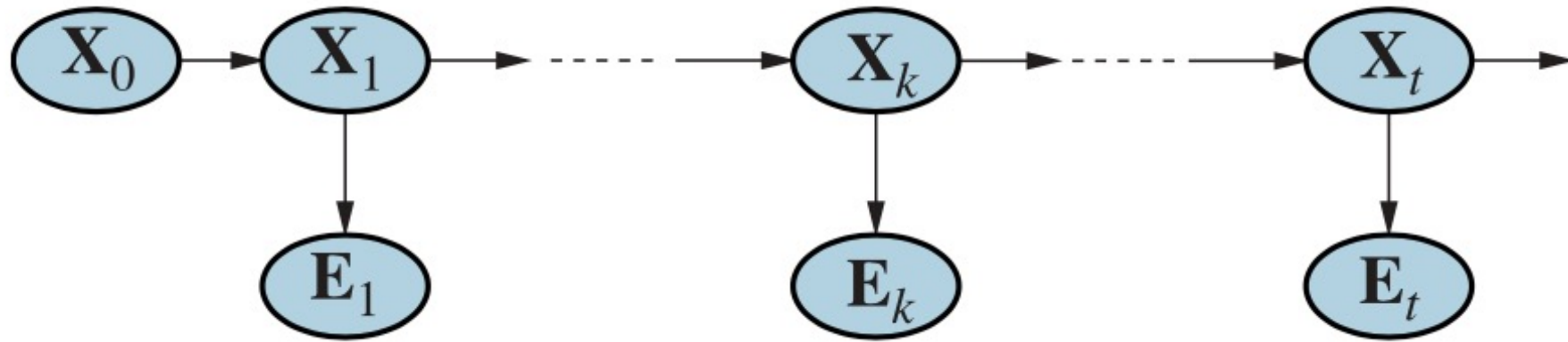


Bayesian Network Structure and Conditional Distributions describing the umbrella world



Smoothing computes $P(X_k | e_{1:t})$

the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t .



The Forward–Backward Algorithm for Smoothing

function FORWARD-BACKWARD(\mathbf{ev} , $prior$) **returns** a vector of probability distributions

inputs: \mathbf{ev} , a vector of evidence values for steps $1, \dots, t$

$prior$, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$

local variables: \mathbf{fv} , a vector of forward messages for steps $0, \dots, t$

\mathbf{b} , a representation of the backward message, initially all 1s

\mathbf{sv} , a vector of smoothed estimates for steps $1, \dots, t$

$\mathbf{fv}[0] \leftarrow prior$

for $i = 1$ **to** t **do**

$\mathbf{fv}[i] \leftarrow \text{FORWARD}(\mathbf{fv}[i - 1], \mathbf{ev}[i])$

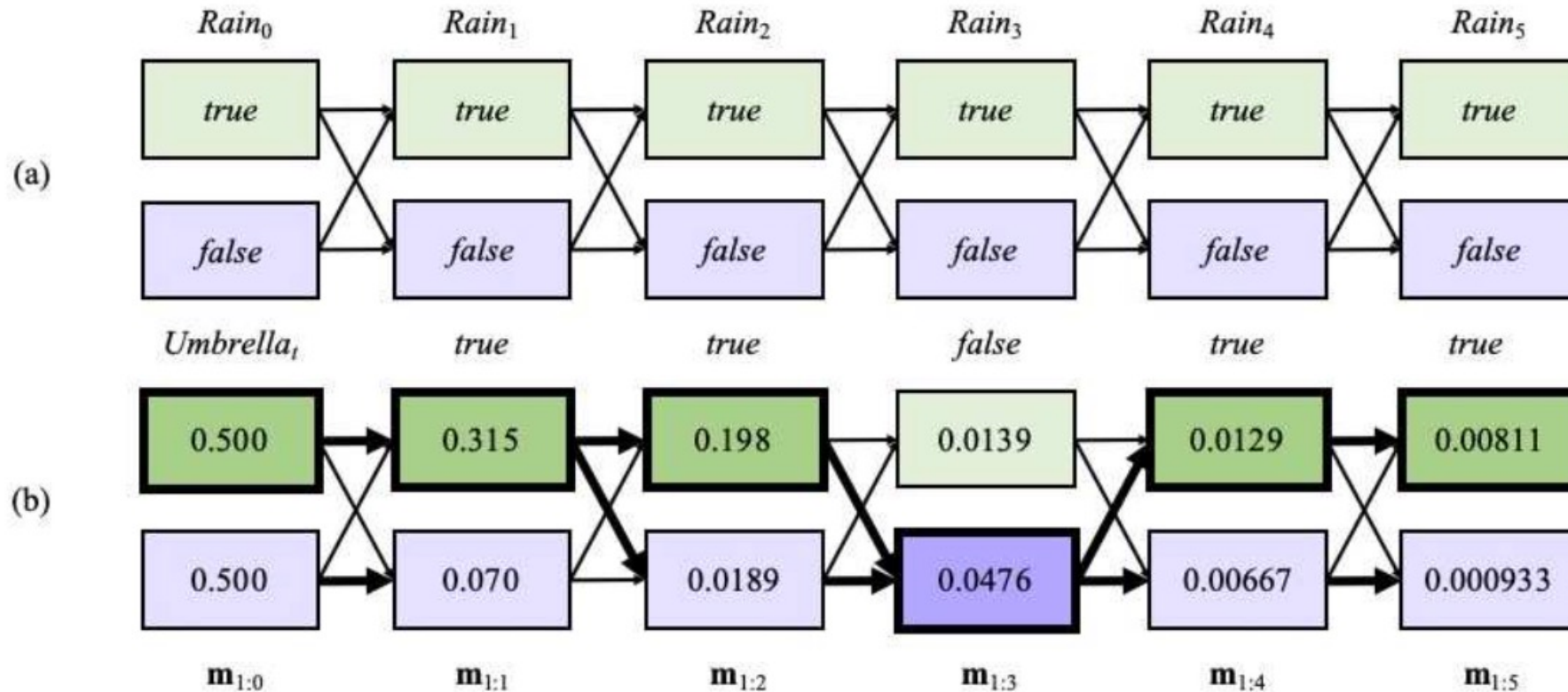
for $i = t$ **down to** 1 **do**

$\mathbf{sv}[i] \leftarrow \text{NORMALIZE}(\mathbf{fv}[i] \times \mathbf{b})$

$\mathbf{b} \leftarrow \text{BACKWARD}(\mathbf{b}, \mathbf{ev}[i])$

return \mathbf{sv}

Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step



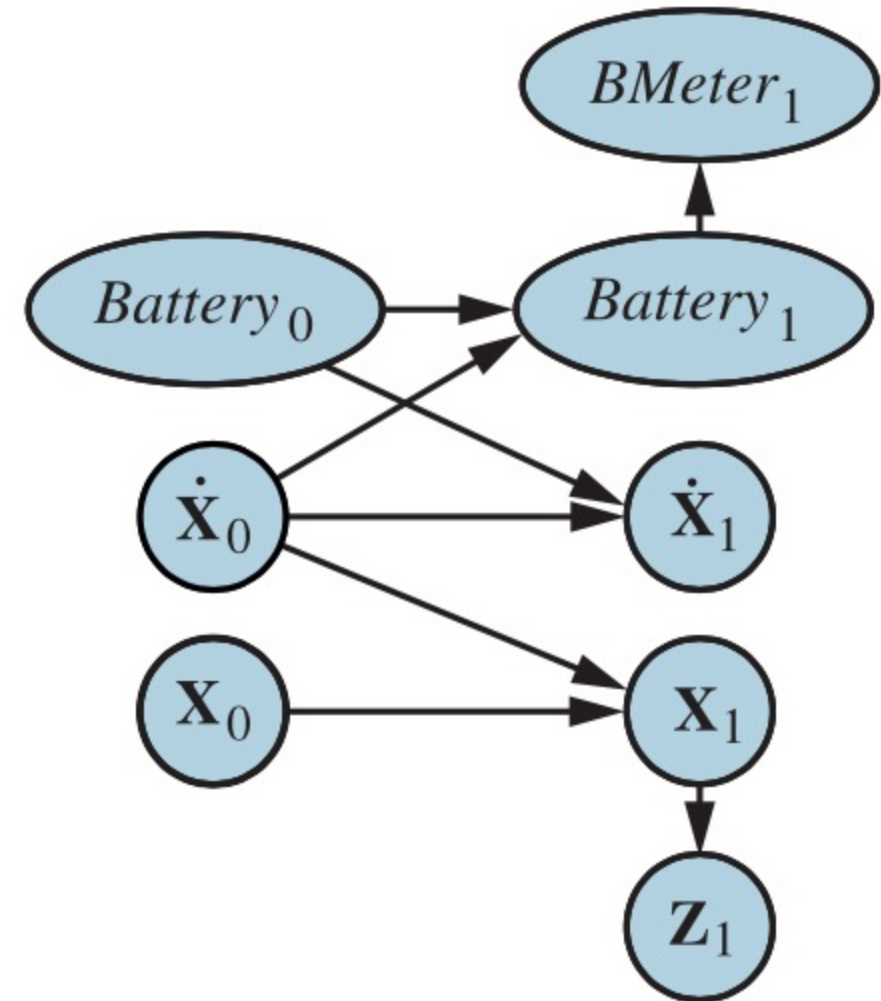
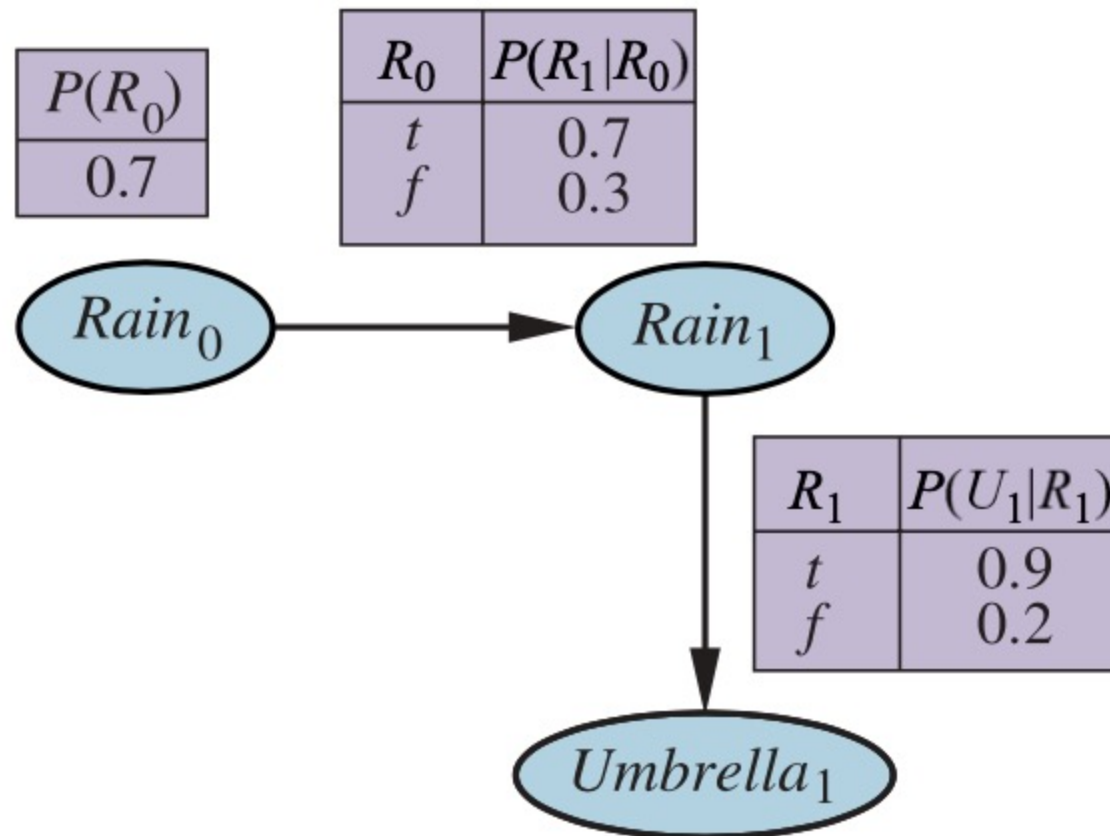
Operation of the Viterbi algorithm
for the umbrella observation sequence $[true, true, false, true, true]$

Algorithm for Smoothing with a Fixed Time Lag of d Step

function FIXED-LAG-SMOOTHING(e_t, hmm, d) **returns** a distribution over \mathbf{X}_{t-d}
inputs: e_t , the current evidence for time step t
 hmm , a hidden Markov model with $S \times S$ transition matrix \mathbf{T}
 d , the length of the lag for smoothing
persistent: t , the current time, initially 1
 \mathbf{f} , the forward message $\mathbf{P}(X_t | e_{1:t})$, initially $hmm.PRIOR$
 \mathbf{B} , the d -step backward transformation matrix, initially the identity matrix
 $e_{t-d:t}$, double-ended list of evidence from $t - d$ to t , initially empty
local variables: $\mathbf{O}_{t-d}, \mathbf{O}_t$, diagonal matrices containing the sensor model information

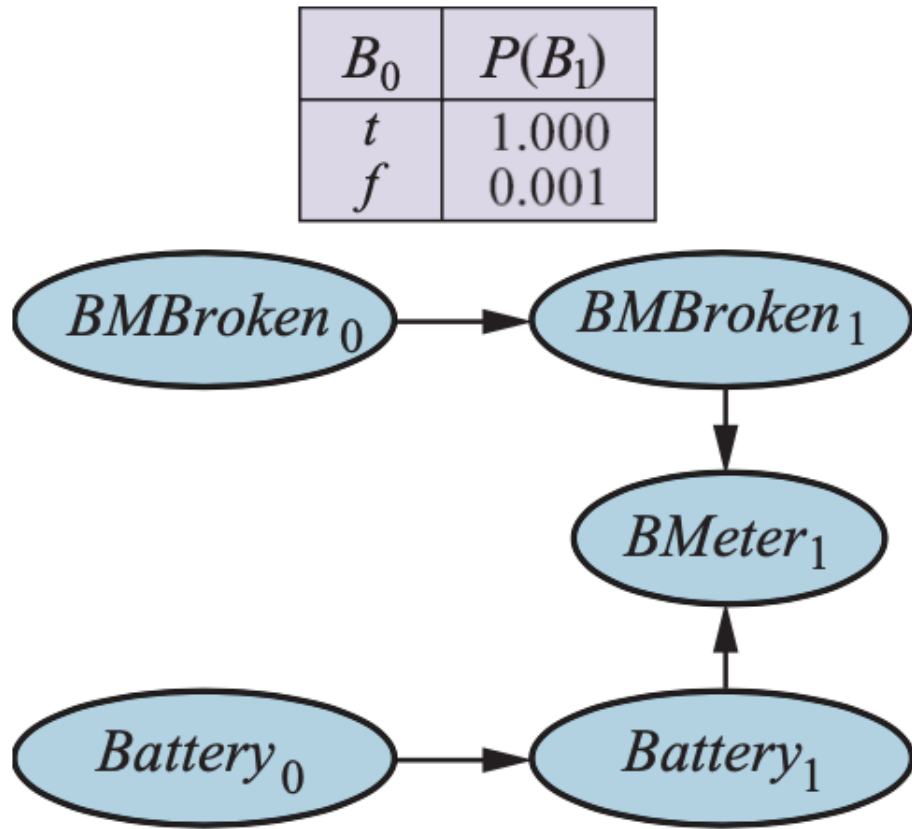
add e_t to the end of $e_{t-d:t}$
 $\mathbf{O}_t \leftarrow$ diagonal matrix containing $\mathbf{P}(e_t | X_t)$
if $t > d$ **then**
 $\mathbf{f} \leftarrow$ FORWARD(\mathbf{f}, e_{t-d})
 remove e_{t-d-1} from the beginning of $e_{t-d:t}$
 $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d} | X_{t-d})$
 $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1} \mathbf{T}^{-1} \mathbf{B} \mathbf{T} \mathbf{O}_t$
else $\mathbf{B} \leftarrow \mathbf{B} \mathbf{T} \mathbf{O}_t$
 $t \leftarrow t + 1$
if $t > d + 1$ **then return** NORMALIZE($\mathbf{f} \times \mathbf{B} \mathbf{1}$) **else return** null

Specification of the prior, transition model, and sensor model for the umbrella DBN

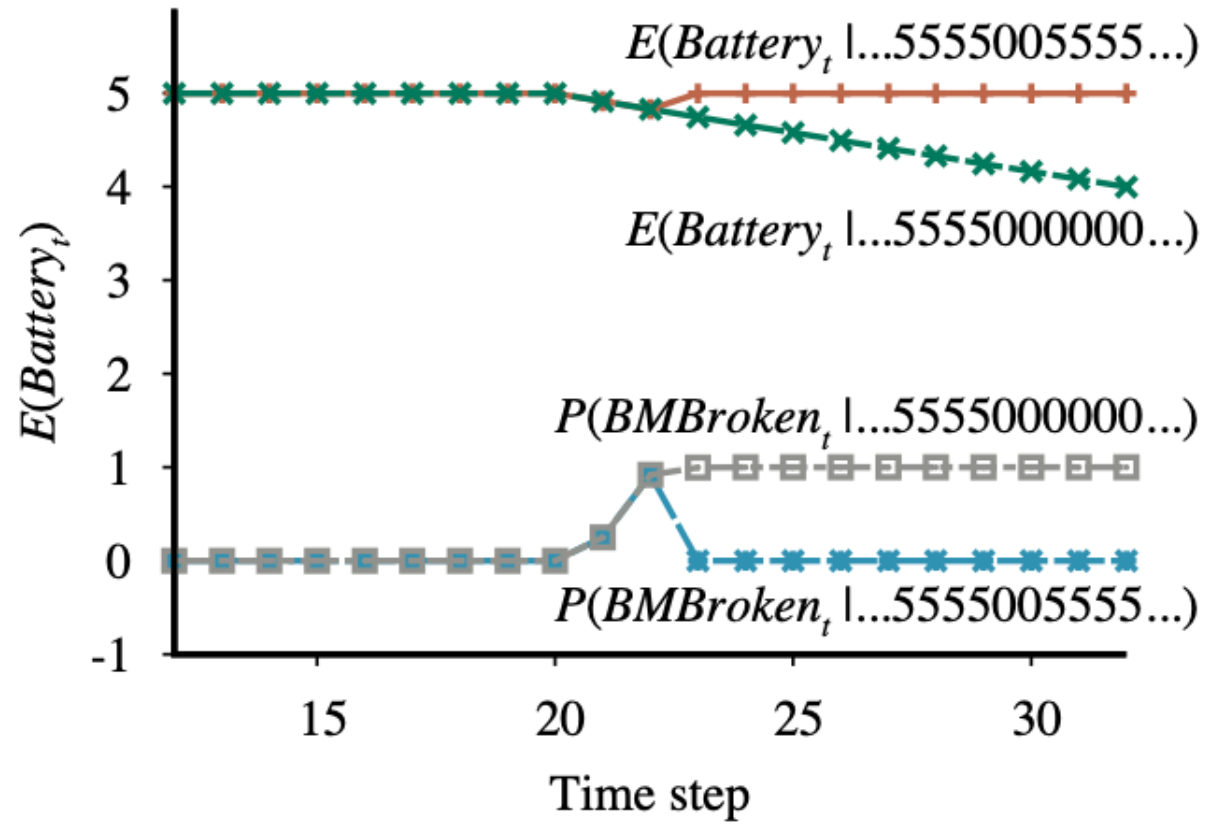


A DBN fragment

the sensor status variable required for modeling persistent failure of the battery sensor

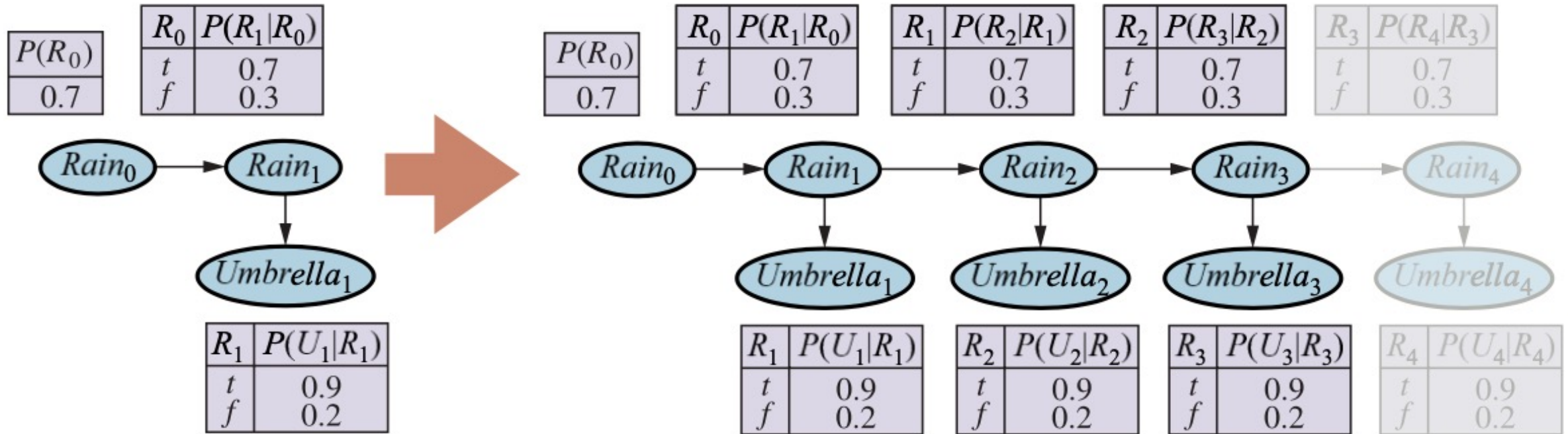


(a)



(b)

Unrolling a Dynamic Bayesian Network



The Particle Filtering Algorithm

function PARTICLE-FILTERING(\mathbf{e} , N , dbn) **returns** a set of samples for the next time step

inputs: \mathbf{e} , the new incoming evidence

N , the number of samples to be maintained

dbn , a DBN defined by $\mathbf{P}(\mathbf{X}_0)$, $\mathbf{P}(\mathbf{X}_1 | \mathbf{X}_0)$, and $\mathbf{P}(\mathbf{E}_1 | \mathbf{X}_1)$

persistent: S , a vector of samples of size N , initially generated from $\mathbf{P}(\mathbf{X}_0)$

local variables: W , a vector of weights of size N

for $i = 1$ to N **do**

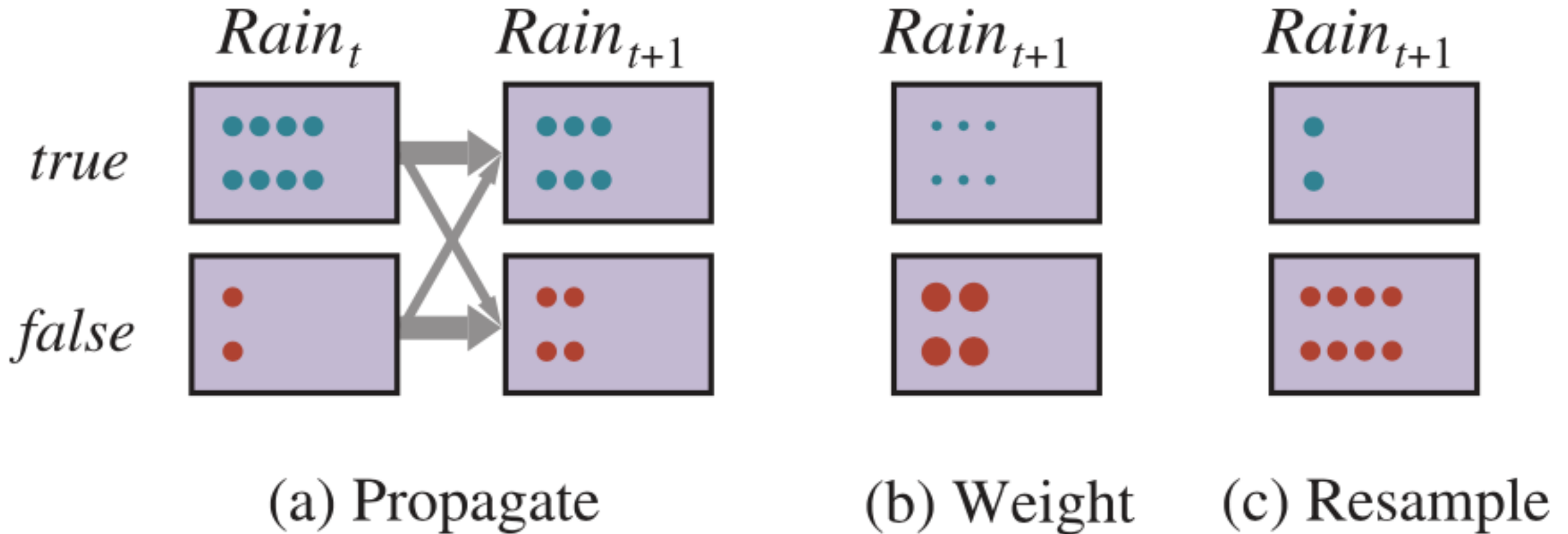
$S[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_1 | \mathbf{X}_0 = S[i])$ // *step 1*

$W[i] \leftarrow \mathbf{P}(\mathbf{e} | \mathbf{X}_1 = S[i])$ // *step 2*

$S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W) // *step 3*

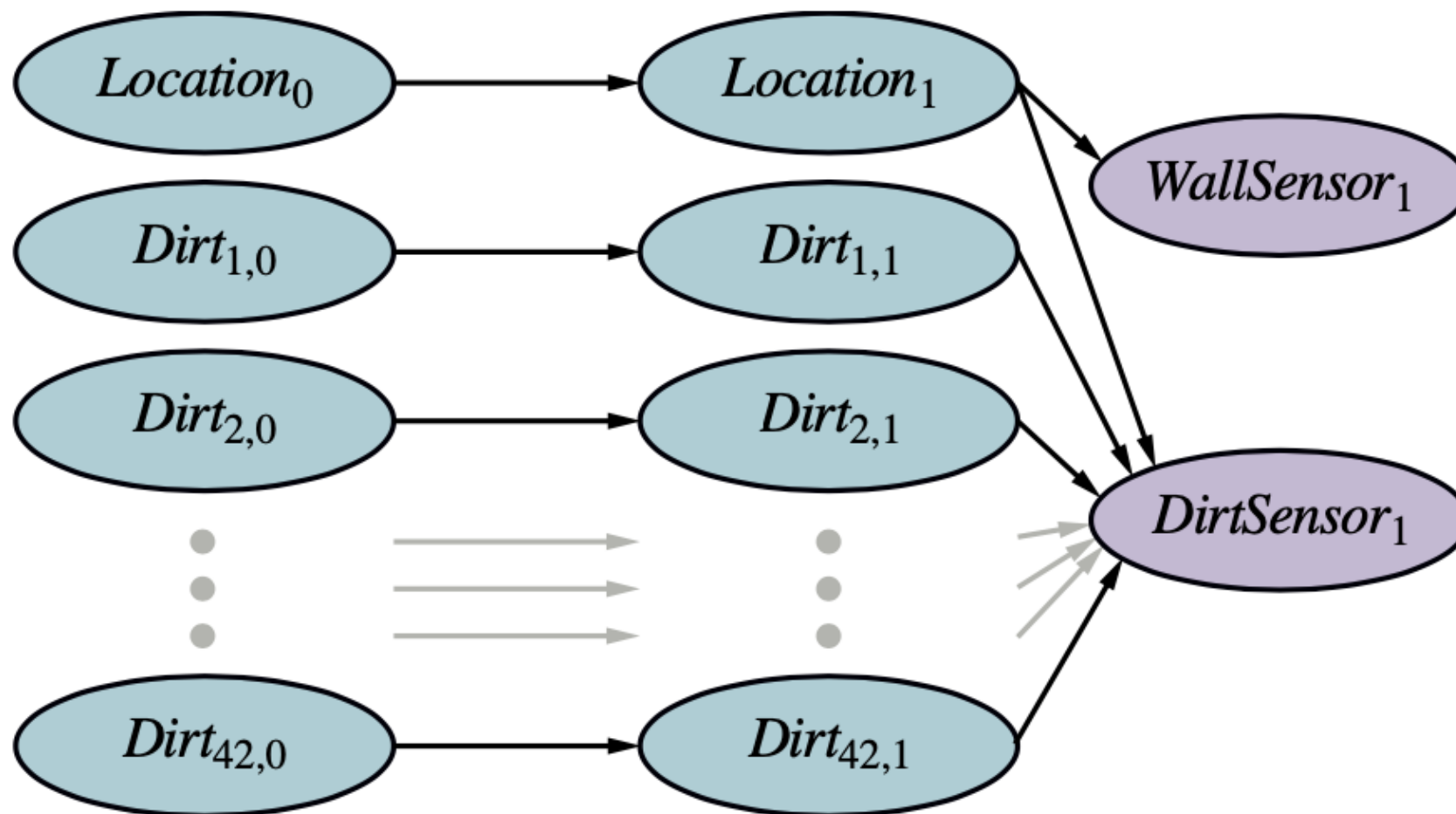
return S

The Particle Filtering Update Cycle for the Umbrella DBN



A Dynamic Bayes Net

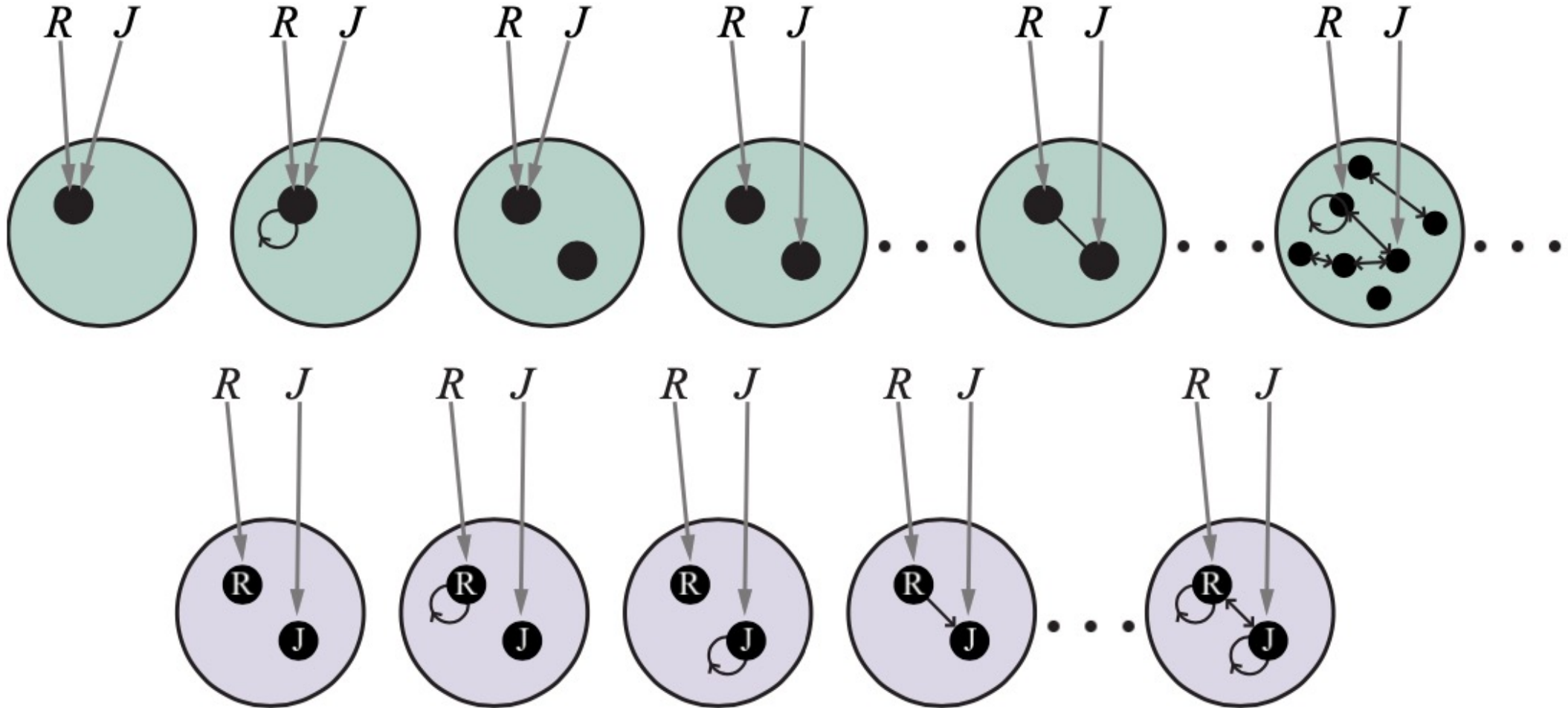
for simultaneous localization and mapping
in the stochastic-dirt vacuum world



Probabilistic Programming

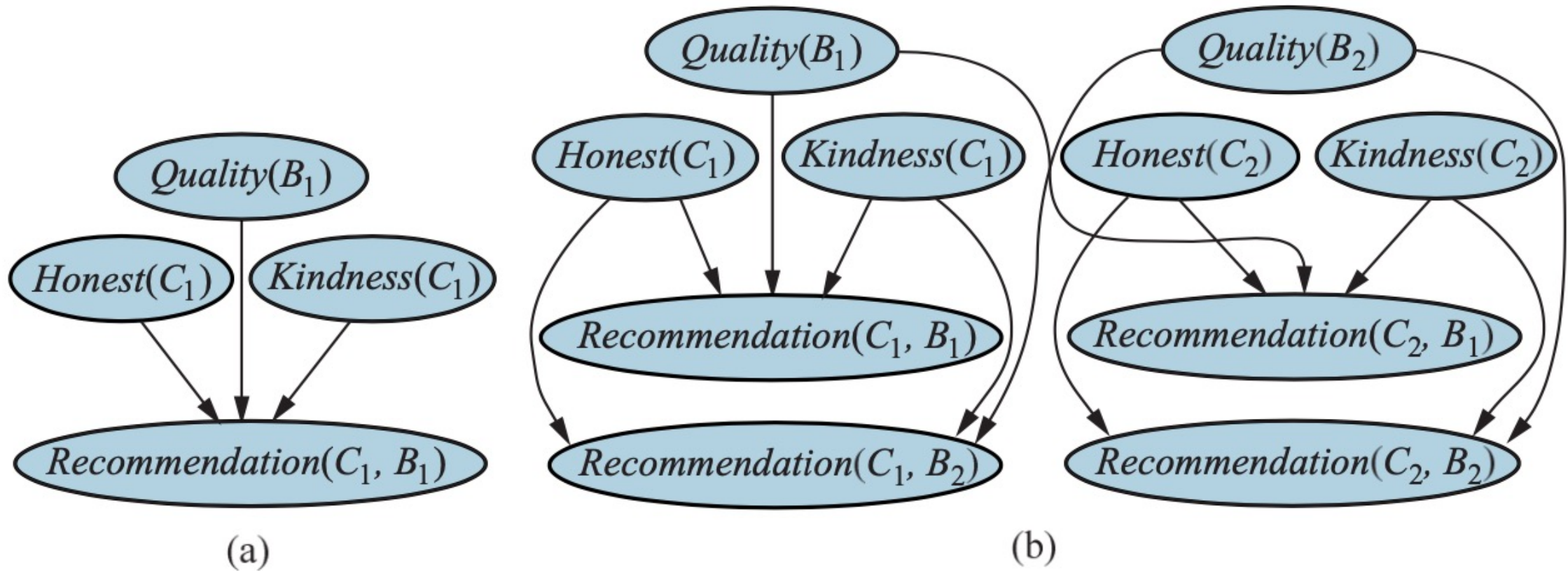
Possible Worlds

for a language with two constant symbols, R and J



Bayes Net for a Single customer C1

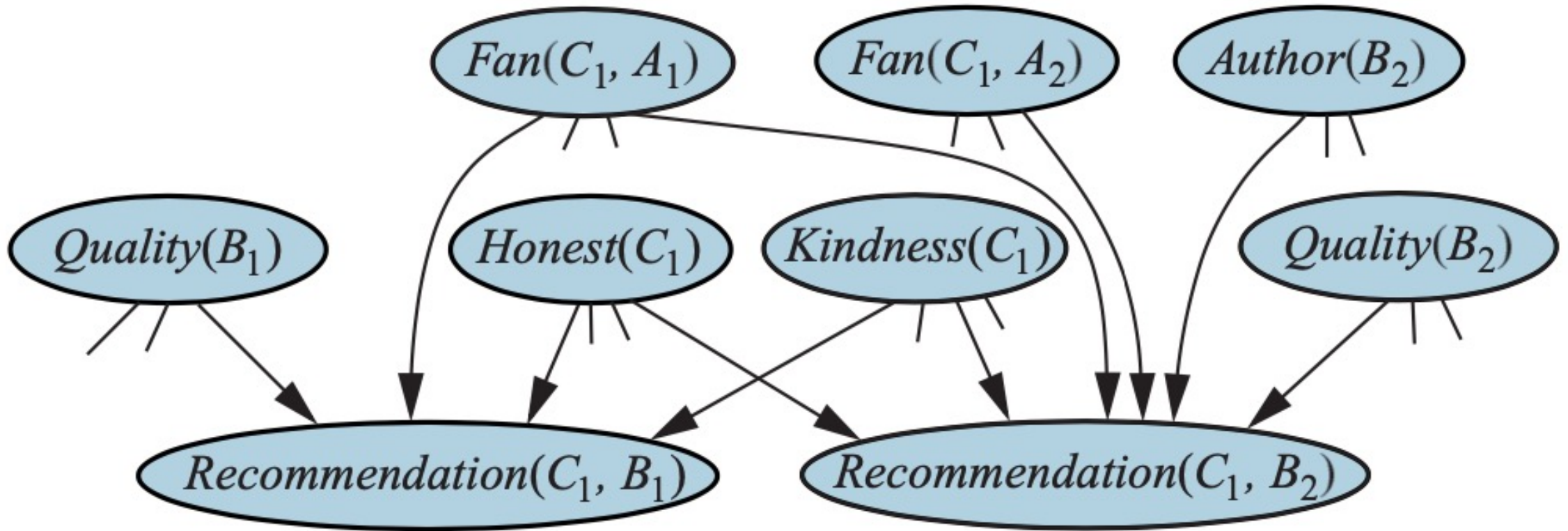
recommending a single book B1. Honest(C1) is Boolean



Bayes net with two customers and two books

Bayes Net

for the book recommendation when Author(B2) is unknown



One particular world for the book recommendation OUPM

Variable	Value	Probability
<i>#Customer</i>	2	0.3333
<i>#Book</i>	3	0.3333
<i>Honest</i> _(Customer, 1)	<i>true</i>	0.99
<i>Honest</i> _(Customer, 2)	<i>false</i>	0.01
<i>Kindness</i> _(Customer, 1)	4	0.3
<i>Kindness</i> _(Customer, 2)	1	0.1
<i>Quality</i> _(Book, 1)	1	0.05
<i>Quality</i> _(Book, 2)	3	0.4
<i>Quality</i> _(Book, 3)	5	0.15
<i>#LoginID</i> _{(Owner, (Customer, 1))}	1	1.0
<i>#LoginID</i> _{(Owner, (Customer, 2))}	2	0.25
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 1)), 1), (Book, 1)}	2	0.5
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 1)), 1), (Book, 2)}	4	0.5
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 1)), 1), (Book, 3)}	5	0.5
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 1), (Book, 1)}	5	0.4
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 1), (Book, 2)}	5	0.4
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 1), (Book, 3)}	1	0.4
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 2), (Book, 1)}	5	0.4
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 2), (Book, 2)}	5	0.4
<i>Recommendation</i> _{(LoginID, (Owner, (Customer, 2)), 2), (Book, 3)}	1	0.4

An OUPM for Citation Information Extraction

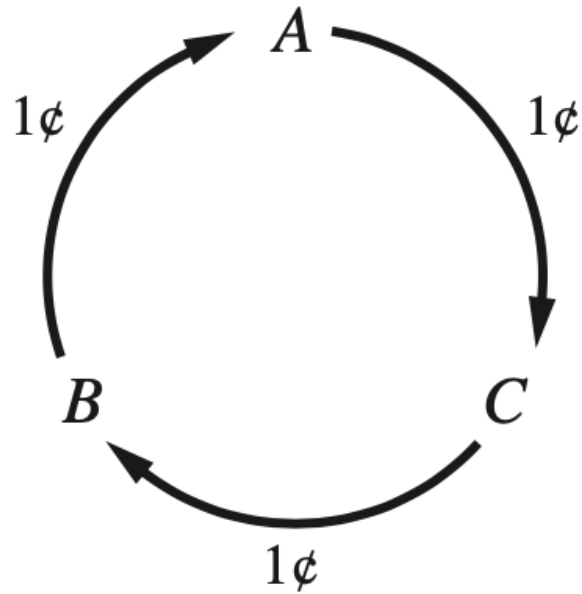
```
type Researcher, Paper, Citation  
random String Name(Researcher)  
random String Title(Paper)  
random Paper PubCited(Citation)  
random String Text(Citation)  
random Boolean Professor(Researcher)  
origin Researcher Author(Paper)
```

```
#Researcher  $\sim$  OM(3, 1)  
Name(r)  $\sim$  NamePrior()  
Professor(r)  $\sim$  Boolean(0.2)  
#Paper(Author = r)  $\sim$  if Professor(r) then OM(1.5, 0.5) else OM(1, 0.5)  
Title(p)  $\sim$  PaperTitlePrior()  
CitedPaper(c)  $\sim$  UniformChoice({Paper p})  
Text(c)  $\sim$  HMMGrammar(Name(Author(CitedPaper(c))), Title(CitedPaper(c)))
```

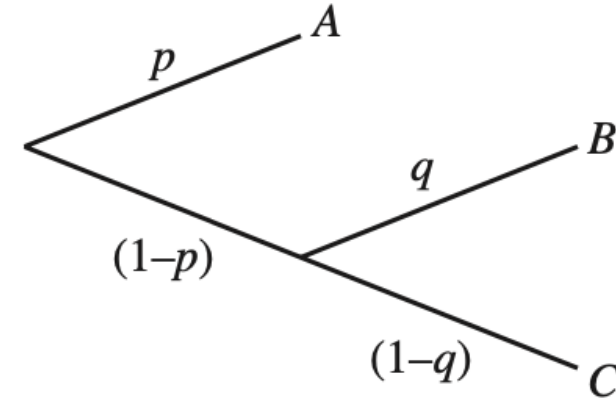
Making Simple Decisions

Nontransitive preferences $A \succ B \succ C \succ A$

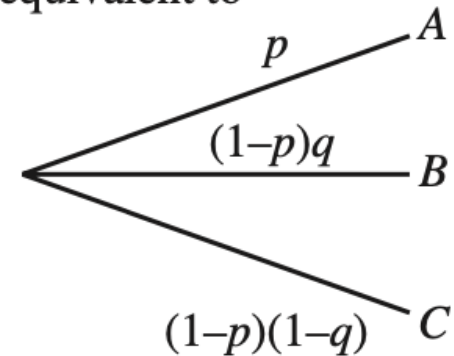
can result in irrational behavior:
a cycle of exchanges each costing one cent



(a)



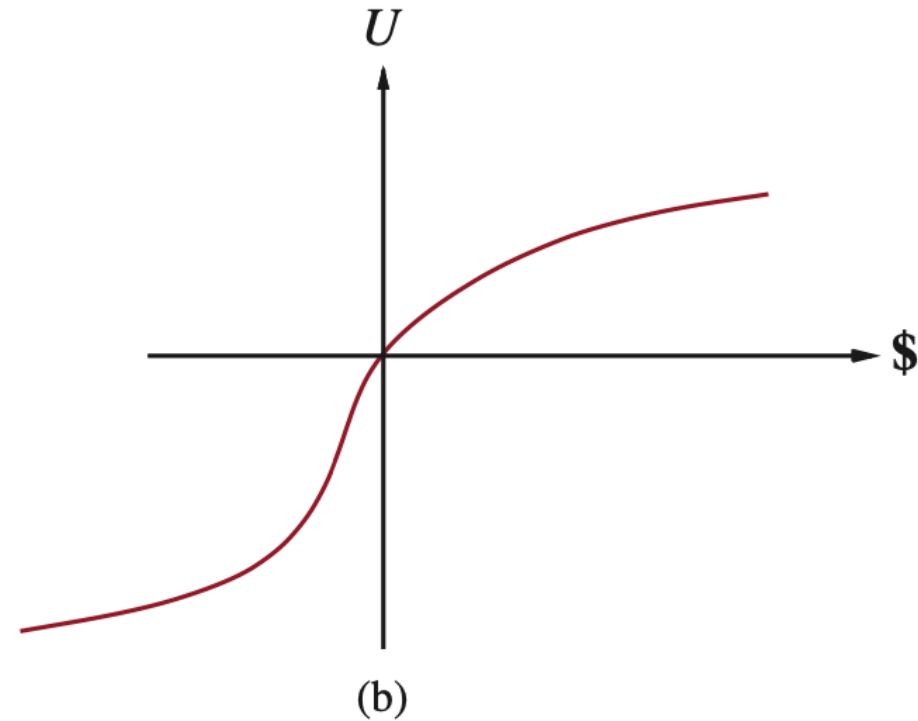
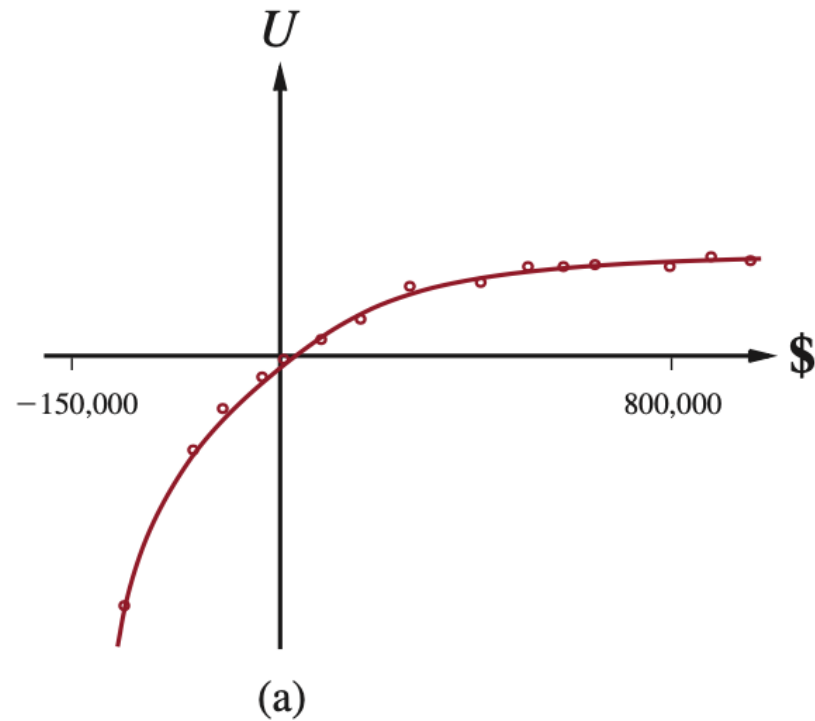
is equivalent to



(b)

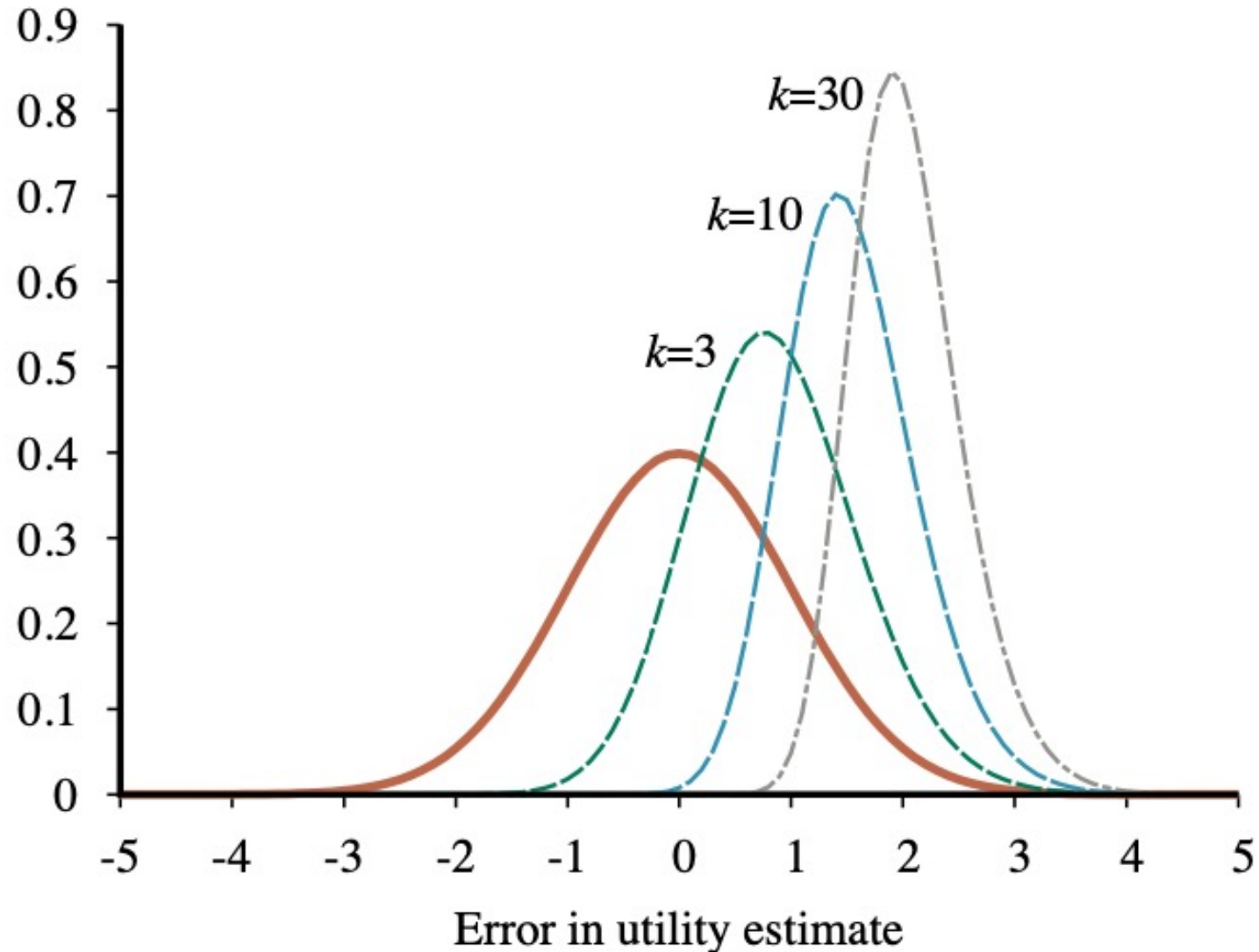
The decomposability axiom

The Utility of Money



Unjustified optimism

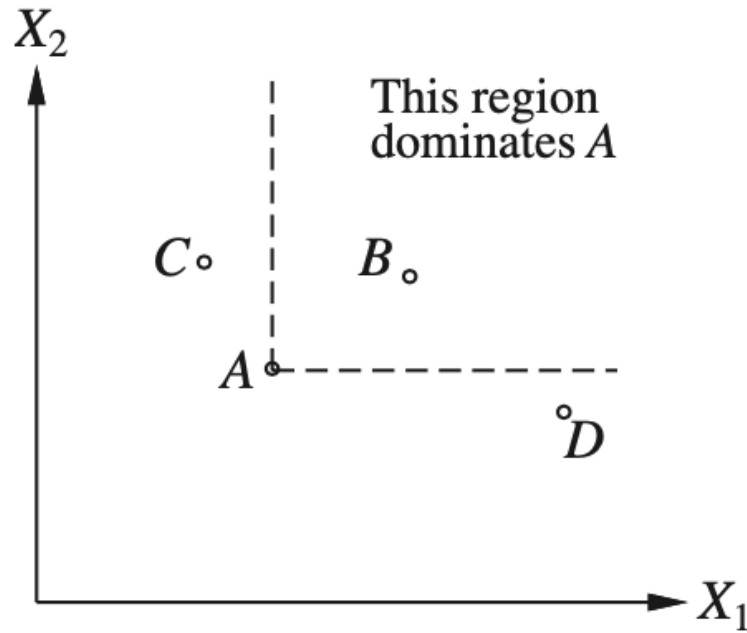
caused by choosing the best of k options



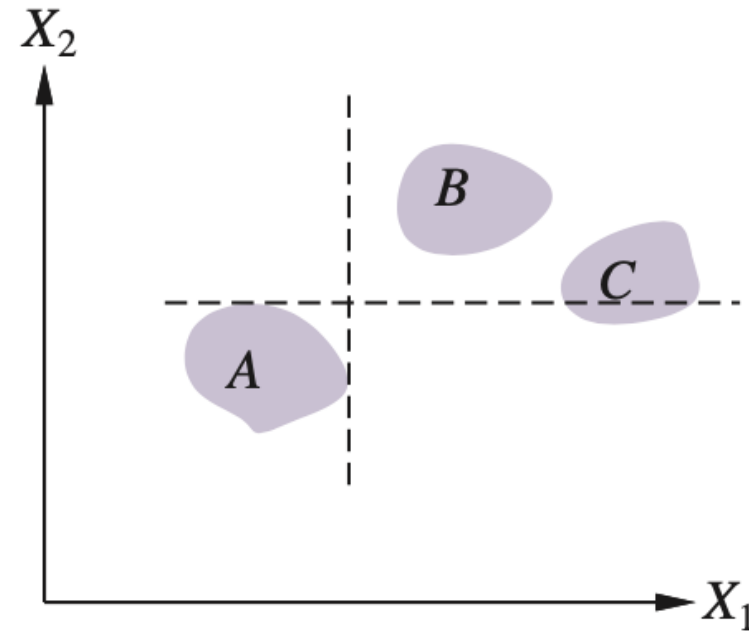
Strict dominance

(a) Deterministic

(b) Uncertain

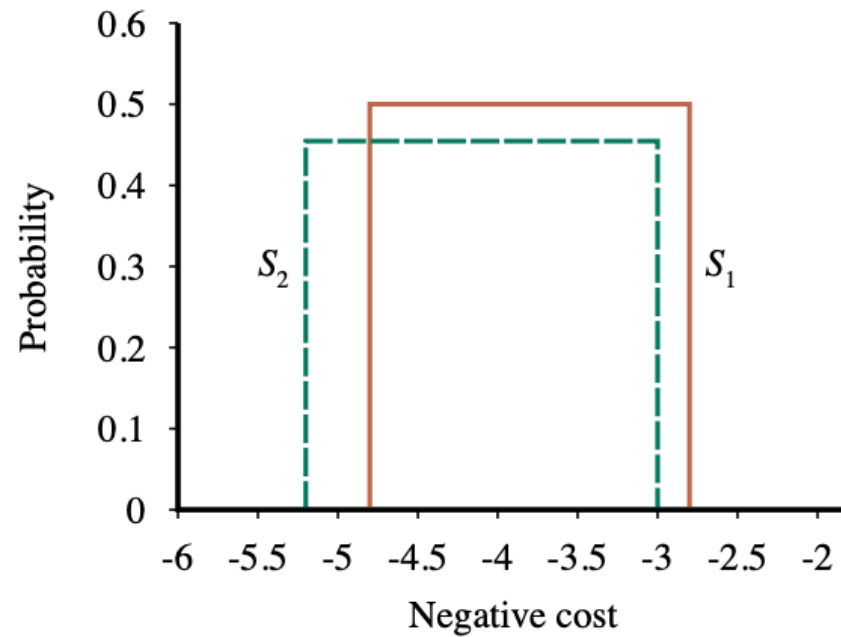


(a)

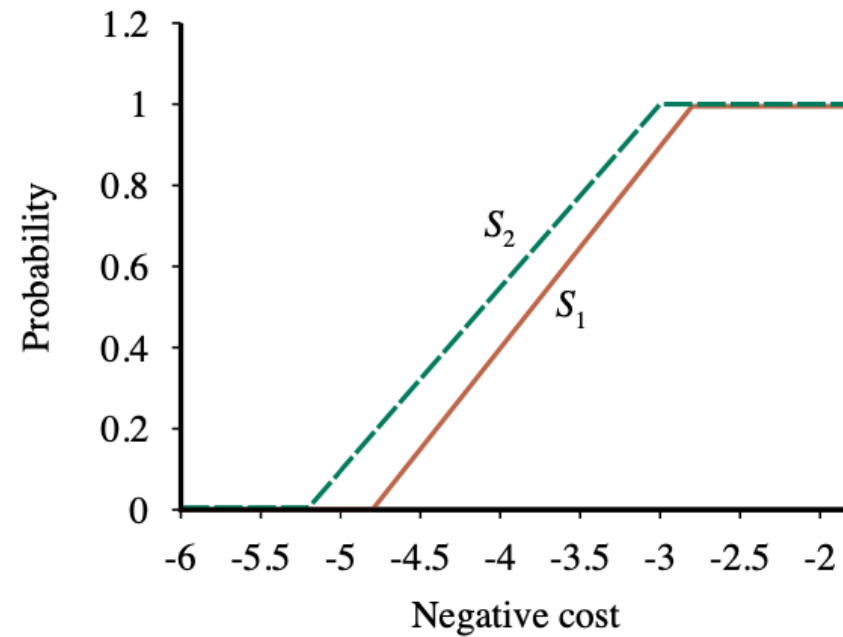


(b)

Stochastic dominance



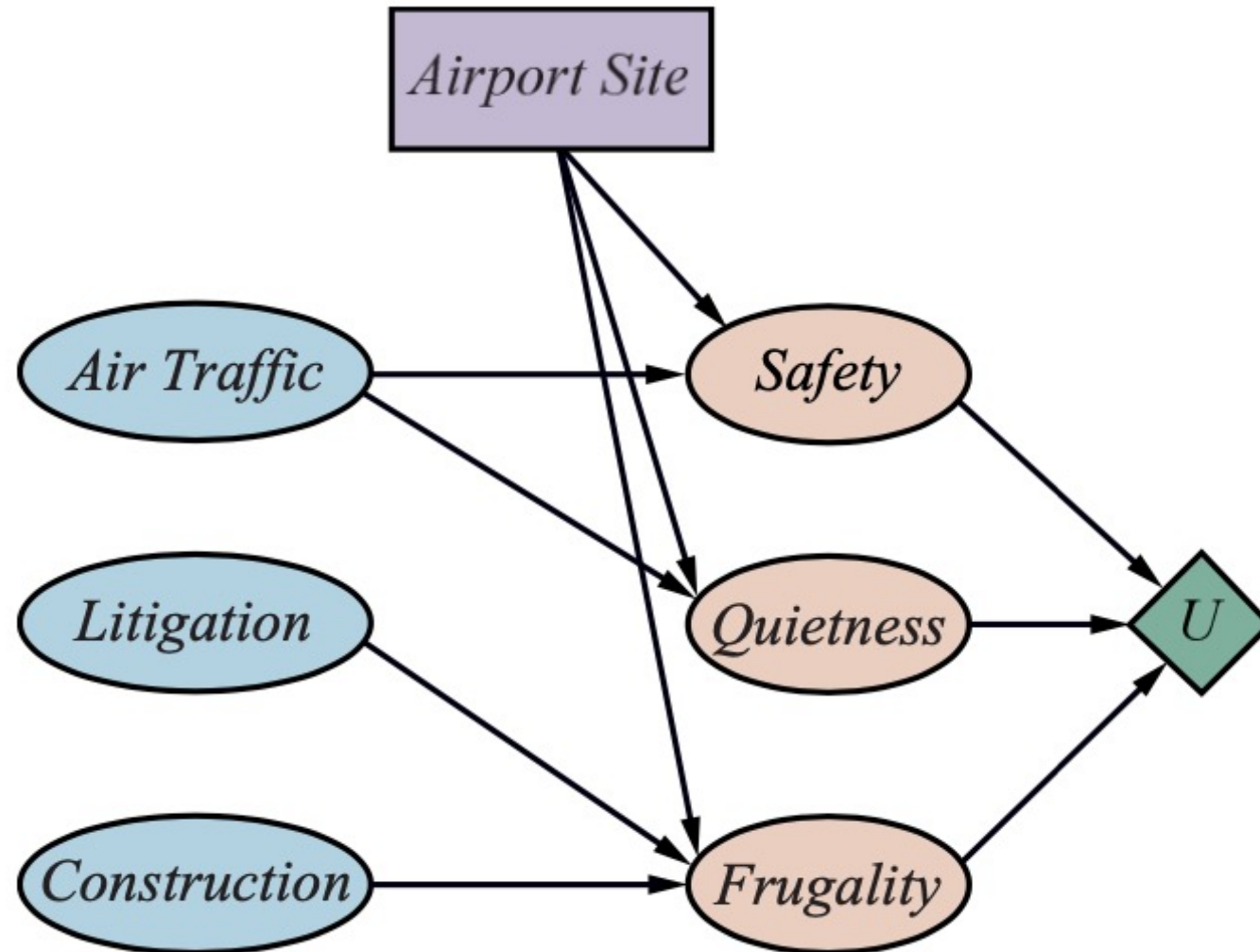
(a)



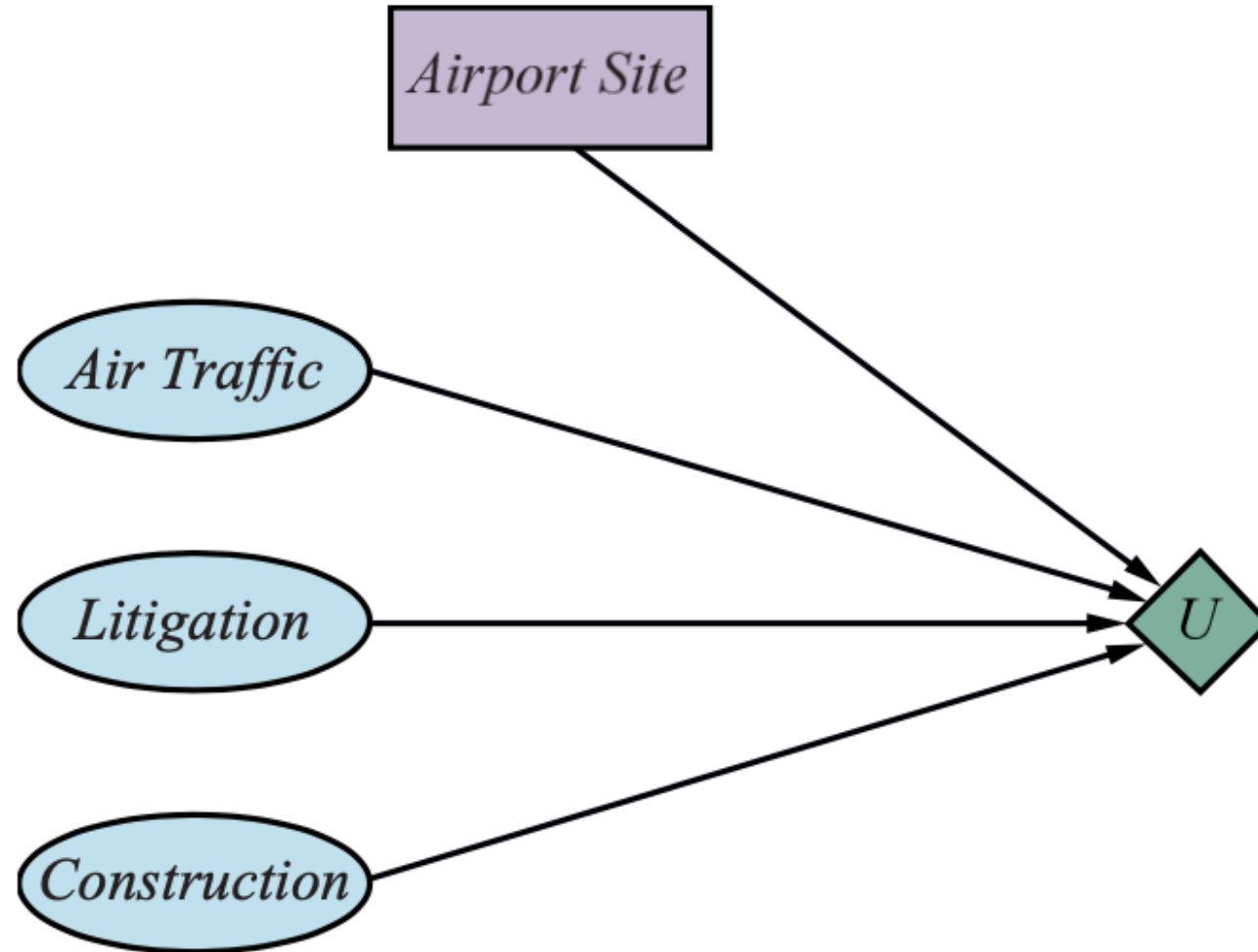
(b)

Cumulative distributions for the frugality of S_1 and S_2 .

A decision network for the airport-siting problem



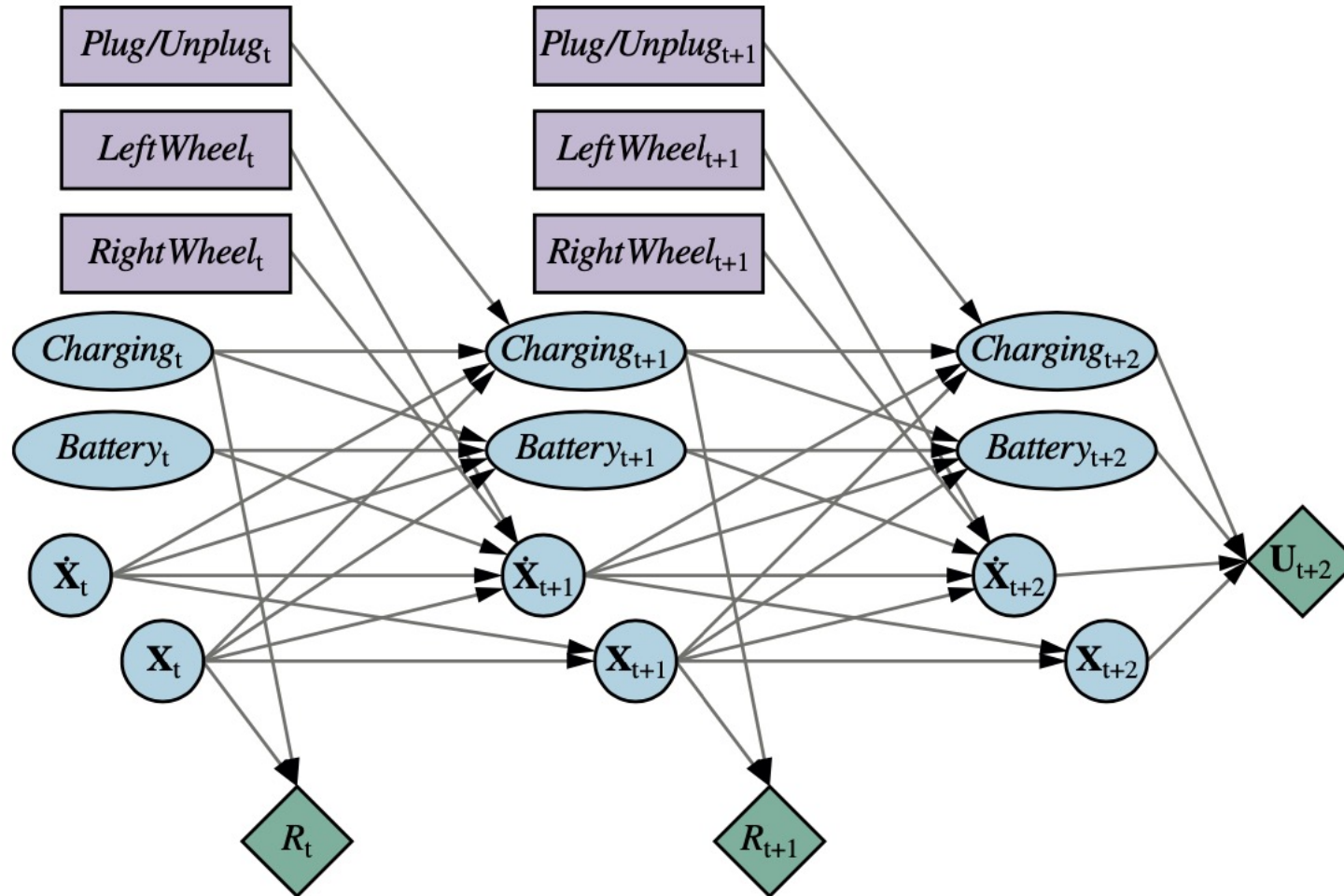
A simplified representation of the airport-siting problem



Making Complex Decisions

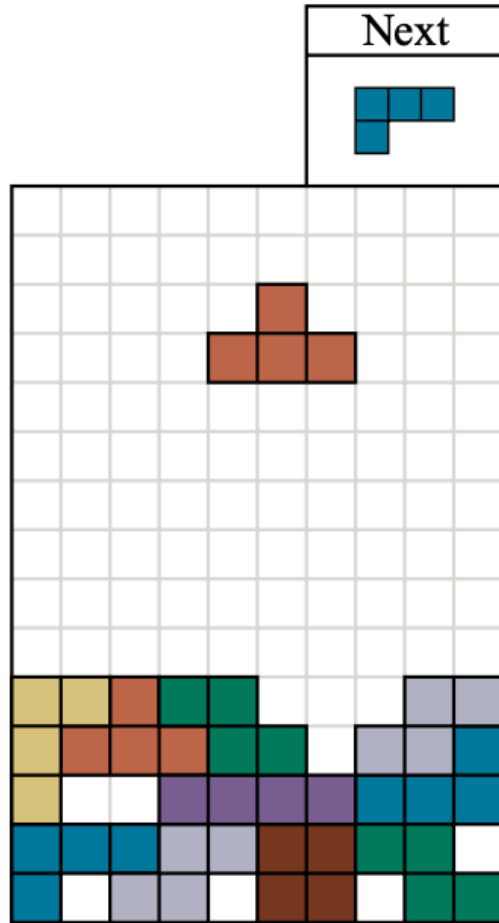
A dynamic decision network

for a mobile robot with state variables for battery level, charging status, location, and velocity, and action variables for the left and right wheel motors and for charging.

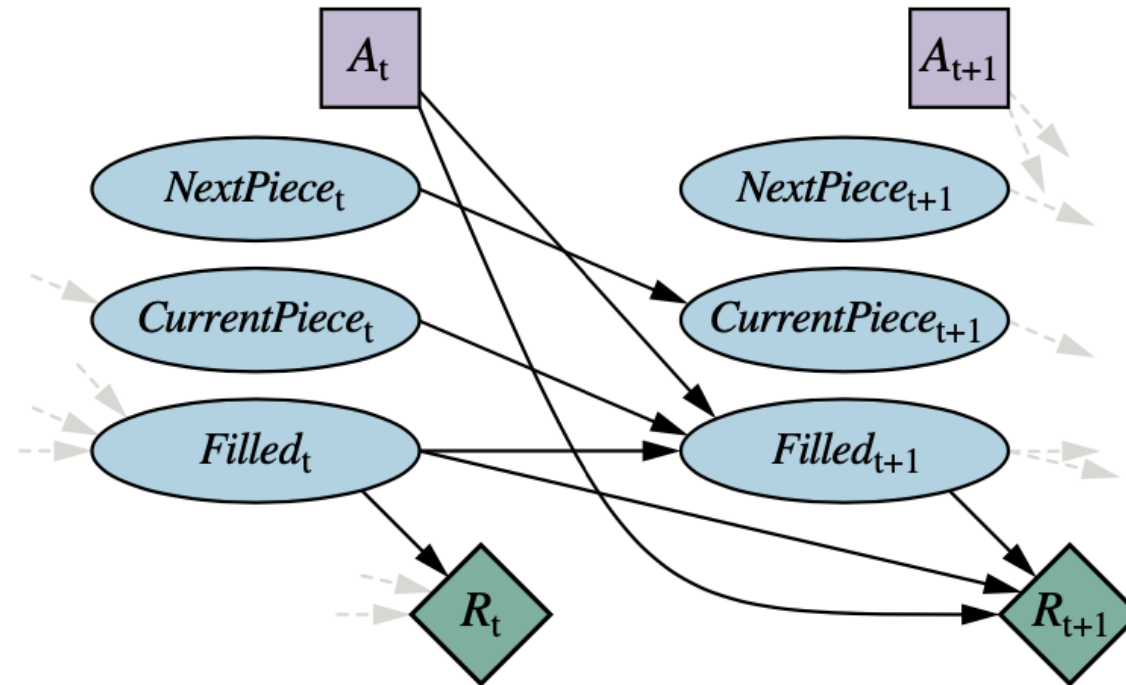


The game of Tetris

The DDN for the Tetris MDP



(a)



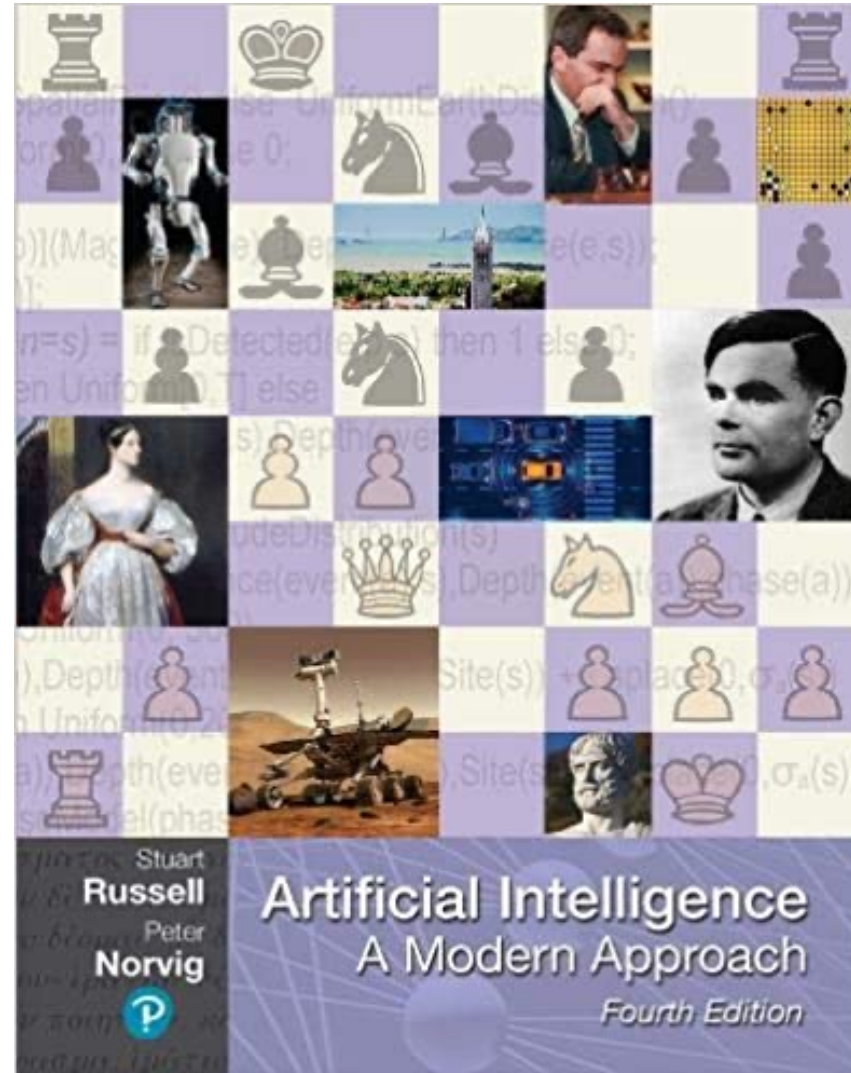
(b)

The Value Iteration Algorithm for calculating utilities of states

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp , an MDP with states S , actions $A(s)$, transition model $P(s' | s, a)$,
rewards $R(s, a, s')$, discount γ
 ϵ , the maximum error allowed in the utility of any state
local variables: U, U' , vectors of utilities for states in S , initially zero
 δ , the maximum relative change in the utility of any state

repeat
 $U \leftarrow U'; \delta \leftarrow 0$
 for each state s **in** S **do**
 $U'[s] \leftarrow \max_{a \in A(s)} \text{Q-VALUE}(mdp, s, a, U)$
 if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta \leq \epsilon(1 - \gamma)/\gamma$
return U

Stuart Russell and Peter Norvig (2020),
Artificial Intelligence: A Modern Approach,
4th Edition, Pearson



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

<https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997/>

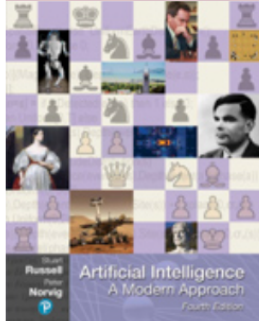
Artificial Intelligence: A Modern Approach (AIMA)

- **Artificial Intelligence: A Modern Approach (AIMA)**
 - <http://aima.cs.berkeley.edu/>
- **AIMA Python**
 - <http://aima.cs.berkeley.edu/python/readme.html>
 - <https://github.com/aimacode/aima-python>
- **Logic, KB Agent**
 - <http://aima.cs.berkeley.edu/python/logic.html>
- **Probability Models (DTAgent)**
 - <http://aima.cs.berkeley.edu/python/probability.html>
- **Markov Decision Processes (MDP)**
 - <http://aima.cs.berkeley.edu/python/mdp.html>

Artificial Intelligence: A Modern Approach (AIMA)



- △ US Edition
- △ Global Edition
- Acknowledgements
- Code
- Courses
- Editions
- Errata
- Exercises
- Figures
- Instructors Page
- Pseudocode
- Reviews



Artificial Intelligence: A Modern Approach, 4th US ed.

by Stuart Russell and Peter Norvig

The authoritative, most-used AI textbook, adopted by over 1500 schools.

Table of Contents for the US Edition (or see the Global Edition)

[Preface \(pdf\)](#); [Contents with subsections](#)

I Artificial Intelligence

- 1 Introduction ... 1
- 2 Intelligent Agents ... 36

II Problem-solving

- 3 Solving Problems by Searching ... 63
- 4 Search in Complex Environments ... 110
- 5 Adversarial Search and Games ... 146
- 6 Constraint Satisfaction Problems ... 180

III Knowledge, reasoning, and planning

- 7 Logical Agents ... 208
- 8 First-Order Logic ... 251
- 9 Inference in First-Order Logic ... 280
- 10 Knowledge Representation ... 314
- 11 Automated Planning ... 344

IV Uncertain knowledge and reasoning

- 12 Quantifying Uncertainty ... 385
- 13 Probabilistic Reasoning ... 412
- 14 Probabilistic Reasoning over Time ... 461
- 15 Probabilistic Programming ... 500
- 16 Making Simple Decisions ... 528
- 17 Making Complex Decisions ... 562
- 18 Multiagent Decision Making ... 599

V Machine Learning

- 19 Learning from Examples ... 651
- 20 Learning Probabilistic Models ... 721
- 21 Deep Learning ... 750
- 22 Reinforcement Learning ... 789

VI Communicating, perceiving, and acting

- 23 Natural Language Processing ... 823
- 24 Deep Learning for Natural Language Processing ... 856
- 25 Computer Vision ... 881
- 26 Robotics ... 925

VII Conclusions

- 27 Philosophy, Ethics, and Safety of AI ... 981
- 28 The Future of AI ... 1012
- Appendix A: Mathematical Background ... 1023
- Appendix B: Notes on Languages and Algorithms ... 1030
- Bibliography ... 1033 ([pdf](#) and [LaTeX .bib file](#) and [bib data](#))
- Index ... 1069 ([pdf](#))

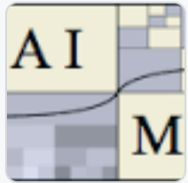
[Exercises \(website\)](#)

[Figures \(pdf\)](#)

[Code \(website\)](#); [Pseudocode \(pdf\)](#)

Covers: [US](#), [Global](#)

AIMA Code



aimacode

Code for the book "Artificial Intelligence: A Modern Approach"

358 followers Berkeley, CA <http://aima.cs.berkeley.edu> peter@norvig.com

Follow

Overview Repositories 13 Projects Packages People

Popular repositories

aima-python

Public

Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

Jupyter Notebook ☆ 6.6k 🍴 3.2k

aima-java

Public

Java implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

Java ☆ 1.4k 🍴 767

aima-pseudocode

Public

Pseudocode descriptions of the algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

☆ 740 🍴 386

aima-exercises

Public

Exercises for the book Artificial Intelligence: A Modern Approach

HTML ☆ 611 🍴 353

aima-javascript

Public

Javascript visualization of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

JavaScript ☆ 495 🍴 208

aima-lisp

Public

Common Lisp implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

Common Lisp ☆ 342 🍴 95

You can now follow organizations

Organization activity like new discussions, sponsorships, and repositories will appear in [your dashboard feed](#).

OK, got it!

members. You must be a member to see who's a part of this organization.

Top languages

JavaScript Python Java
Common Lisp Scala

Report abuse

<https://github.com/aimacode>

AIMA Python

aimacode / aima-python Public

Watch 337

Fork 3.2k

Star 6.6k

Code Issues 120 Pull requests 79 Actions Projects Wiki Security Insights

master

1 branch 0 tags

Go to file

Add file

Code



mcventur Fixed bug in treatment of repeated nodes in frontier... 61d695b on Dec 5, 2021 1,190 commits



aima-data @ f6cbea6 updating submodule (#994) 4 years ago



gui fixed tests (#1191) 2 years ago



images add perception and tests (#1091) 3 years ago



js Added TicTacToe to notebook (#213) 7 years ago



notebooks Image Rendering problem resolved (#1178) 3 years ago



tests fixed tests (#1191) 2 years ago



.coveragerc Added coverage report generation to Travis (#1058) 3 years ago



.flake8 Fix flake8 warnings (#508) 5 years ago



.gitignore Reworked PriorityQueue and Added Tests (#1025) 4 years ago



.gitmodules Updating Submodule (#647) 5 years ago



.travis.yml fixed svm for not posdef kernel matrix, updated .travis.yml wi... 2 years ago

About

Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

Readme

MIT license

6.6k stars

337 watching

3.2k forks

Releases

No releases published

Packages

No packages published

Papers with Code State-of-the-Art (SOTA)



Search for papers, code and tasks



[Browse State-of-the-Art](#)

[Follow](#)

[Discuss](#)

[Trends](#)

[About](#)

[Log In/Register](#)

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on [Twitter](#) for updates

Computer Vision



Semantic Segmentation

33 leaderboards
667 papers with code



Image Classification

52 leaderboards
564 papers with code



Object Detection

54 leaderboards
467 papers with code



Image Generation

51 leaderboards
231 papers with code



Pose Estimation

40 leaderboards
231 papers with code

[See all 707 tasks](#)

Natural Language Processing



Machine Translation



Language Modelling



Question Answering



Sentiment Analysis

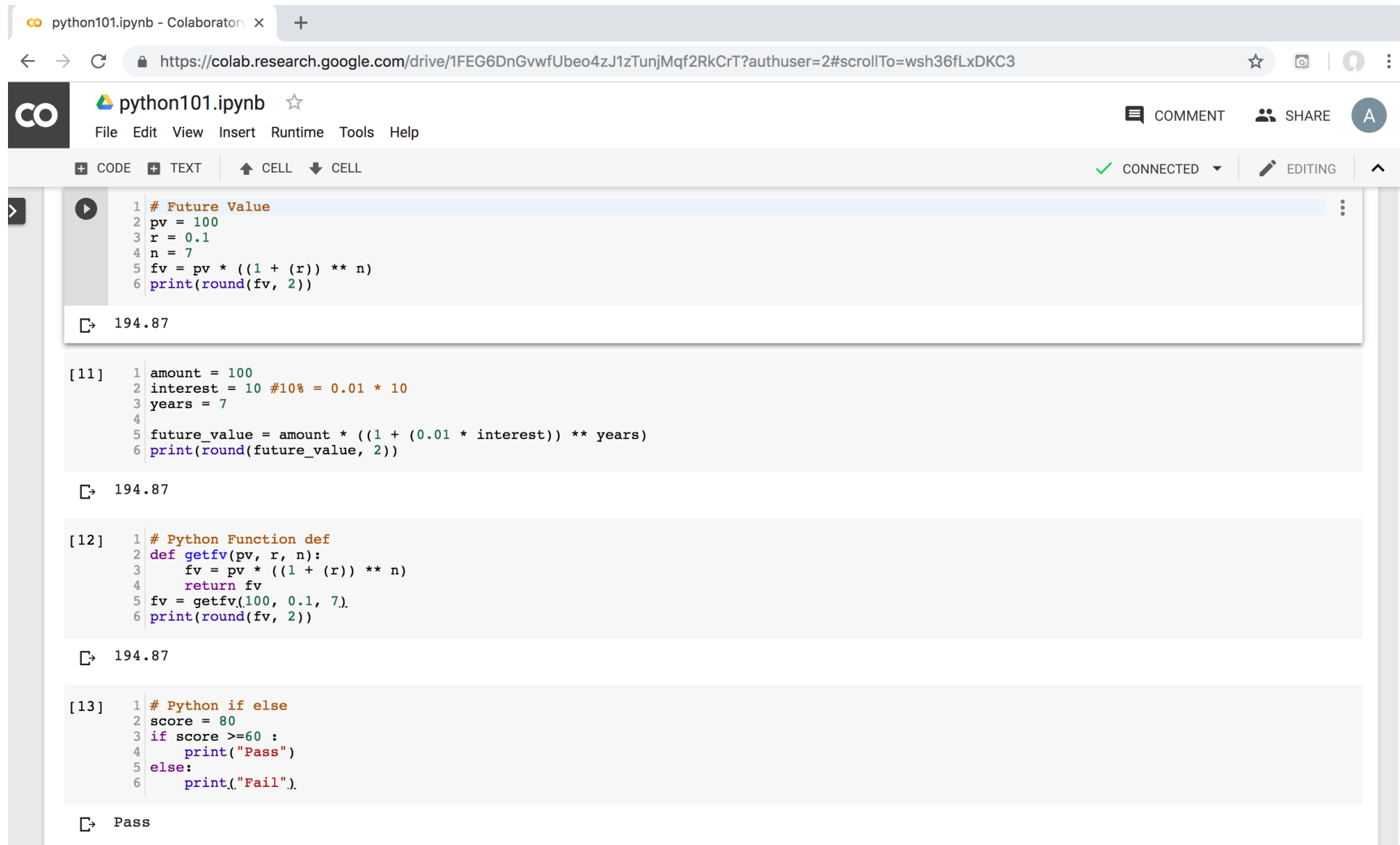


Text Generation

<https://paperswithcode.com/sota>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. On the right, there are buttons for "COMMENT", "SHARE", and a user profile icon. Below the navigation bar, there are tabs for "CODE", "TEXT", "CELL", and "CELL", along with a "CONNECTED" status indicator and an "EDITING" mode button.

The notebook contains four code cells, each followed by its output:

```
1 # Future Value
2 pv = 100
3 r = 0.1
4 n = 7
5 fv = pv * ((1 + (r)) ** n)
6 print(round(fv, 2))
```

194.87

```
[11] 1 amount = 100
2 interest = 10 #10% = 0.01 * 10
3 years = 7
4
5 future_value = amount * ((1 + (0.01 * interest)) ** years)
6 print(round(future_value, 2))
```

194.87

```
[12] 1 # Python Function def
2 def getfv(pv, r, n):
3     fv = pv * ((1 + (r)) ** n)
4     return fv
5 fv = getfv(100, 0.1, 7)
6 print(round(fv, 2))
```

194.87

```
[13] 1 # Python if else
2 score = 80
3 if score >=60 :
4     print("Pass")
5 else:
6     print("Fail").
```

Pass

<https://tinyurl.com/aintpupython101>

Summary

- **Knowledge and Reasoning**
 - **Logical Agents**
 - **First-Order Logic**
 - **Inference in First-Order Logic**
 - **Knowledge Representation**
 - **Knowledge Graph (KG)**
- **Uncertain Knowledge and Reasoning**
 - **Quantifying Uncertainty**
 - **Probabilistic Reasoning**
 - **Making Complex Decisions**

References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Numa Dhamani and Maggie Engler (2024), Introduction to Generative AI, Manning
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Aurélien Géron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Nithin Buduma, Nikhil Buduma, Joe Papa (2022), Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 2nd Edition, O'Reilly Media.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. (2024) "Reasoning with Large Language Models, a Survey." arXiv preprint arXiv:2407.11511.
- Madaan, Aman, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon et al. (2024) "Self-refine: Iterative refinement with self-feedback." Advances in Neural Information Processing Systems 36.