Propositional Logic

10S3001 – Artificial Intelligence

Samuel I. G. Situmeang

Faculty of Informatics and Electrical Engineering

Objectives

Students are able:

- to explain the concept of knowledge-based agents clearly.
- to explain the concept of logical agents clearly.
- to represent information from the environment in the form of logical propositions.
- to apply inference to propositional logic by using propositional logic inference rules.

Siswa mampu:

- untuk menjabarkan konsep knowledge-based agent dengan jelas.
- untuk menjabarkan konsep logical agent dengan jelas.
- untuk merepresentasi informasi dari lingkungan dalam bentuk logika proposisi.
- untuk menerapkan inferensi pada logika proposisi dengan menggunakan aturan inferensi logika proposisi.

Overview

- Knowledge-based Agents
- The Wumpus World
- Logical Agent
- Building Propositions
- Inference Rules
- Model Checking and Inference
- Theorem Proving and Proof by Resolution
- Conversion to CNF and Resolution Algorithm
- Forward and Backward Chaining

Knowledge-based Agents

Knowledge-based Agents

Logical AI:
"The idea is that an agent can represent knowledge of its world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals."

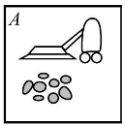
John McCarthy in Concepts of logical AI, 2000.
http://www-formal.stanford.edu/jmc/concepts-ai/concepts-ai.html

- Bayangkan sebuah robot yang ingin mengambil segelas air.
- Untuk mencapai tujuan ini, robot perlu:
  - Memahami dunia: Robot perlu mengetahui di mana dapur berada, di mana air berada, dan bagaimana cara menuju ke sana.
  - Memiliki tujuan: Tujuan robot adalah mengambil segelas air.
  - Bernalar secara logis: Robot perlu berpikir, "Jika saya pergi ke dapur, saya dapat menemukan air. Jika saya menemukan air, saya dapat mengisi gelas. Oleh karena itu, saya harus pergi ke dapur."
- Ini adalah ide dasar dari *Logical* AI dimana AI menggunakan logika untuk merepresentasikan pengetahuan, tujuan, dan tindakan. Dengan bernalar secara logis, AI dapat memutuskan apa yang harus dilakukan untuk mencapai tujuannya.

PS Agent and KB Agent

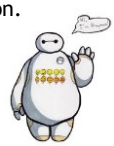
Problem Solving Agent

- choose a solution among the **available** possibilities.
- The knowledge is very **specific and inflexible**. What he "knew" about the world **does not evolve**.
- problem solution (initial state, successor function, goal test).



Knowledge-based Agent

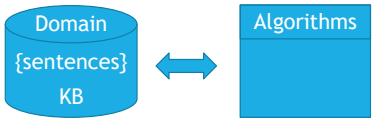
- "smarter".
- The knowledge is very **flexible**.
- What he "knew" about the world **evolve**.
- can do reasoning for:
  - imperfect/partial information.
  - choosing better action.



- *Problem Solving Agent* dirancang khusus untuk menemukan solusi untuk masalah yang diberikan.
- *Knowledge Based Agent* memiliki basis pengetahuan yang luas dan mendalam tentang suatu domain sehingga mampu melakukan penalaran untuk mengambil keputusan berdasarkan pengetahuan yang dimilikinya.

Knowledge-based Agents

- Intelligent agents need **knowledge** about the world to choose good actions/decisions.
- Knowledge = **{sentences}** in a knowledge representation language (formal language).
- A sentence is an assertion about the world.
- A knowledge-based agent is composed of:
  1. Knowledge base: **domain-specific** content.
  2. Inference mechanism: **domain-independent** algorithms



Knowledge-based Agents

- The agent **must be able to**:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions
- **Declarative** approach to building an agent:
  - Add new sentences: **Tell** it what it needs to know
  - Query what is known: **Ask** itself what to do – answers should follow from the KB

Knowledge-based Agents

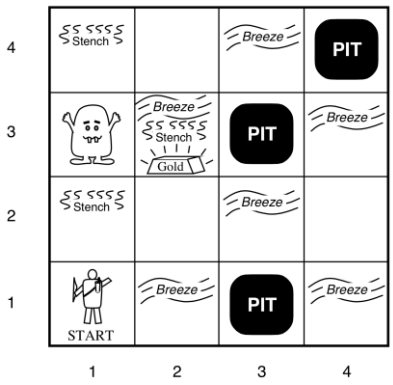
```
function KB-Agent(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  Tell(KB, Make-Percept-Sentence(percept, t))
  action ← Ask(KB, Make-Action-Query(t))
  Tell(KB, Make-Action-Sentence(action, t))
  t ← t + 1
  return action
```

The Wumpus World

The Wumpus World

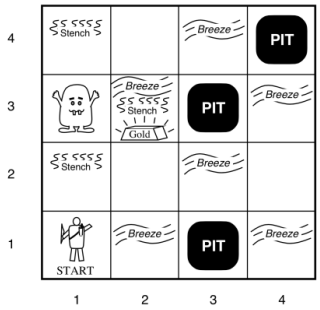
Gregory Yob (1975)



Lingkungan Dunia Wumpus adalah sebuah lingkungan buatan yang sering digunakan sebagai contoh dalam pembelajaran kecerdasan buatan, khususnya dalam bidang pencarian dan perencanaan. Lingkungan ini menyajikan sebuah dunia sederhana namun kompleks, di mana seorang agen (biasanya disebut sebagai agen) harus menjelajahi gua untuk menemukan emas sambil menghindari jebakan berupa lubang (pits) dan makhluk berbahaya seperti Wumpus.

The Wumpus World

- 4 X 4 grid of rooms
- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills Wumpus if you are facing it
- Wumpus emits a horrible scream when it is killed that can be heard anywhere
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square



Wumpus World PEAS

- **Performance measure**:
  - gold +1000, death (eaten or falling in a pit) -1000, -1 per action taken, -10 for using the arrow.
  - The games ends either when the agent dies or comes out of the cave.
- **Environment**
  - 4 X 4 grid of rooms
  - Agent starts in square [1,1] facing to the right
  - Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1]
  - Each square other than the start can be a pit with probability of 0.2
- **Actuators**:
  - Left turn, Right turn, Forward, Grab, Release, Shoot
- **Sensors**:
  - Stench, Breeze, Glitter, Bump, Scream
  - Represented as a 5-element list
  - Example: [Stench, Breeze, None, None, None]

Wumpus World Environment

- Partially observable – only local perception
- Static – Wumpus and Pits do not move
- Discrete
- Single-agent – Wumpus is essentially a natural feature
- Deterministic – outcomes exactly specified
- Sequential – sequential at the level of actions

Karakteristik Lingkungan Dunia Wumpus:

1.Dapat diamati sebagian – hanya persepsi lokal:

1. Agen hanya dapat mengamati lingkungan sekitarnya secara terbatas.
2. Ia tidak memiliki peta lengkap dari seluruh gua.
3. Informasi yang diperoleh hanya berasal dari persepsi langsung terhadap ruangan yang sedang ditempati, seperti adanya bau busuk (mengindikasikan keberadaan Wumpus), adanya tiupan angin (mengindikasikan keberadaan lubang), atau keberadaan emas.

2.Statis – Wumpus dan Pits tidak bergerak:

1. Letak Wumpus dan lubang tidak berubah selama agen menjelajahi gua.
2. Ini menyederhanakan masalah karena agen tidak perlu mempertimbangkan pergerakan objek-objek tersebut.

3.Diskrit:

1. Lingkungan terbagi menjadi ruangan-ruangan yang terpisah.
2. Agen hanya dapat berpindah dari satu ruangan ke ruangan lain melalui pintu yang menghubungkannya.

4.Agen tunggal:

1. Hanya ada satu agen yang berinteraksi dengan lingkungan.
2. Wumpus dianggap sebagai bagian dari lingkungan, bukan sebagai agen

lain yang memiliki tujuan sendiri.

5. Deterministik:

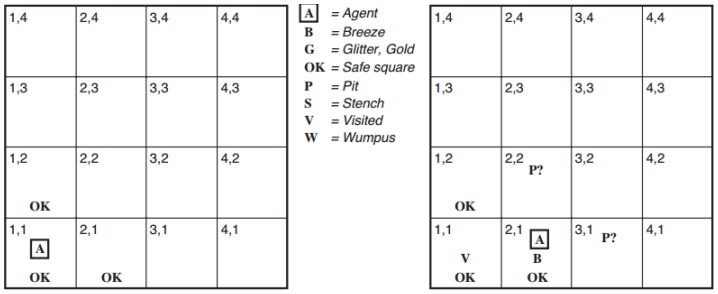
1. Hasil dari setiap tindakan agen dapat diprediksi dengan pasti.
2. Jika agen menembakkan panah ke ruangan sebelah, maka panah akan mengenai sasaran dengan pasti jika ada Wumpus di ruangan tersebut.

6.Berurutan – berurutan pada tingkat tindakan:

1. Tindakan agen dilakukan secara berurutan.
2. Hasil dari satu tindakan akan mempengaruhi tindakan selanjutnya.

Exploring Wumpus World

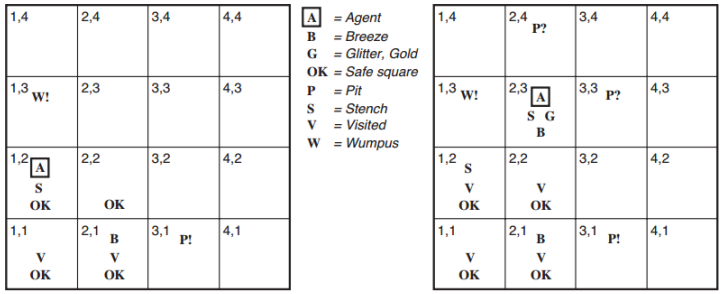
- Agent's first steps:



(a) (b)

Exploring Wumpus World

- Agent's later steps:



(a) (b)

Logical Agent

Logic

- **Knowledge base**: a set of sentences in a formal representation, logic
- **Logics**: are formal languages for representing knowledge to extract conclusions
  - **Syntax**: defines well-formed sentences in the language
    - x + 2 ≥ y **is a sentence**
    - x2 + y > **is not a sentence**
  - **Semantic**: defines the truth or meaning of sentences in a world
    - x + 2 ≥ y **is true iff the number** x + 2 **is no less than the number** y
    - x + 2 ≥ y **is true in a world where** x = 7; y = 1
    - x + 2 ≥ y **is false in a world where** x = 0; y = 6

- Seperti yang telah disebutkan sebelumnya, agen logis membutuhkan basis pengetahuan. Basis pengetahuan tersebut adalah sekumpulan kalimat dalam representasi formal yang disebut logika.
- Logika merupakan bahasa formal untuk merepresentasikan pengetahuan untuk mengekstrak atau menarik kesimpulan.
- Sintaks dan semantik adalah dua konsep fundamental dalam logika yang saling berkaitan erat. Keduanya berperan penting dalam membangun dan memahami struktur serta makna dari pernyataan-pernyataan logistik.

Logic

- **Inference**: a procedure to derive a new sentence from other ones.
- **Logical entailment**: is a relationship between sentences. It means that a sentence follows logically from other sentences.

Knowledge base KB entails sentence α

if and only if

α is true in all worlds where KB is true

Notation:

$$KB \models \alpha$$

- **Logical Entailment** atau **Implikasi Logis** dalam logika adalah sebuah hubungan antara dua pernyataan atau lebih, di mana kebenaran dari satu pernyataan (premis) secara logis menjamin kebenaran pernyataan lainnya (konklusi). Dengan kata lain, jika premis benar, maka konklusi *harus* benar.

Entailment

- **Example**

  The KB containing “the shirt is green” and “the shirt is striped” entails “the shirt is green or the shirt is striped”

- **Example**

  x + y = 4 entails 4 = x + y

Propositional Logic

- Propositional logic (PL) is the simplest logic.
- **Syntax of PL**: defines the allowable sentences or propositions.
- **Definition (Proposition)**: A proposition is a declarative statement that's either `True` or `False`.
  - **Atomic proposition**: single proposition symbol. Each symbol is a proposition. Notation: upper case letters and may contain subscripts.
  - **Compound proposition**: constructed from atomic propositions using parentheses and logical connectives.

Atomic Proposition

- **Examples of atomic propositions:**
  - $2 + 2 = 4$ is a true proposition
  - $W_{1,3}$ is a proposition. It is true if there is a Wumpus in [1,3]
  - "If there is a stench in [1,2] then there is a Wumpus in [1,3]" is a proposition
  - "How are you?" or "Hello!" are not propositions. In general, statement that are questions, commands, or opinions are not propositions.

Compound Proposition

- **Examples of compound/complex propositions:**

  Let p, p_1, and p_2 be propositions

  - **Negation** : $\neg p$ is also a proposition. We call a **literal** either an atomic proposition or its negation. E.g., $W_{1,3}$ is a positive literal, and $\neg W_{1,3}$ is a negative literal.
  - **Conjunction** $p_1 \wedge p_2$. E.g., $W_{1,3} \wedge P_{3,1}$
  - **Disjunction** $p_1 \vee p_2$. E.g., $W_{1,3} \vee P_{3,1}$
  - **Implication** $p_1 \Rightarrow p_2$. E.g., $W_{1,3} \wedge P_{3,1} \Rightarrow \neg W_{2,2}$
  - **If and only if** $p_1 \Leftrightarrow p_2$. E.g., $W_{1,3} \Leftrightarrow \neg W_{2,2}$

Truth Tables

- The semantics define the rules to determine the truth of a sentence.
- Semantics can be specified by truth tables.
- Boolean values domain: T, F
- n-tuple: $(x_1, x_2, \ldots, x_n)$
- Operator on n-tuples : $g(x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n)$
- Definition: A truth table defines an operator g on n-tuples by specifying a boolean value for each tuple
- Number of rows in a truth table? $\mathrm{R} = 2^n$

The number of rows in the truth table is 2^n, where n is a number of atomic propositions.

Building Propositions

Building Propositions

- **Negation:**

p	$\neg p$
T	F
F	T

- **Conjunction:**

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Building Propositions

- **Disjunction:**

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

- **Exclusive or:**

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Building Propositions

- **Implication:**

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

- **Biconditional or If and only if (IFF):**

p	q	$p \Leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Precedence of Operators

- Just like arithmetic operators, there is an operator precedence when evaluating logical operators as follows:
  1. Expressions in parentheses are processed (inside to outside)
  2. Negation
  3. AND
  4. OR
  5. Implication
  6. Biconditional
  7. Left to right
- Use parentheses whenever you have any doubt!

Building Propositions

p	q	r	$\neg r$	$p \vee q$	$p \vee q \Rightarrow \neg r$
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

Logical Equivalence

- Two propositions p and q are logically equivalent if and only if the columns in the truth table giving their truth values agree.
- We write this as $p \Leftrightarrow q$ or $p \equiv q$.

p	q	$\neg p$	$\neg p \vee q$	$p \Rightarrow q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Properties of Operators

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

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

Tautology and Contradiction

- **Tautology** is a proposition which is always true
- **Contradiction** is a proposition which is always false
- **Contingency** is a proposition which is neither a tautology or a contradiction

p	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
T	F	T	F
F	T	T	F

Contrapositive, Inverse, etc.

- Given an **implication** $p \Rightarrow q$
- The **converse** is: $q \Rightarrow p$
- The **contrapositive** is: $\neg q \Rightarrow \neg p$
- The **inverse** is: $\neg p \Rightarrow \neg q$

Contrapositive, Inverse, etc.

- Given an **implication** $p \Rightarrow q$
- The **converse** is: $q \Rightarrow p$
- The **contrapositive** is: $\neg q \Rightarrow \neg p$
- The **inverse** is: $\neg p \Rightarrow \neg q$

Example: Hot is a sufficient condition for my going to the beach.

- The **implication** is: $hot \Rightarrow beach$
- The **converse** is: $beach \Rightarrow hot$
- The **contrapositive** is: $\neg beach \Rightarrow \neg hot$
- The **inverse** is: $\neg hot \Rightarrow \neg beach$

Among the connectives, the implication plays a crucial role in inference. Let's see first it's different variants and also clarify the terminology.
So first of all, an implication is a Boolean expression, or actually, a proposition of the form p implies q. So sometimes you would see it written as p implies q, but this is the same.

Inference Rules

Inference (Modus Ponens)

$$\frac{p \qquad p \Rightarrow q}{q}$$

$$\frac{warm \qquad warm \Rightarrow sunny}{sunny}$$

Inference (Modus Ponens)

- **Horn clauses**: a proposition of the form:

$$p_1 \wedge \cdots \wedge p_n \Rightarrow q$$

- Modus Ponens deals with **Horn clauses**:

$$\frac{p_1, \ldots, p_n \qquad (p_1 \wedge \cdots \wedge p_n) \Rightarrow q}{q}$$

Modus Ponens can be extended by handling **what we call Horn clauses It's a proposition of the form p1,** and p2, and pn implies q.

Inference (Modus Tollens)

$$\frac{\neg q \qquad p \Rightarrow q}{\neg p}$$

$$\frac{\neg beach \qquad hot \Rightarrow beach}{\neg hot}$$

Common Rules

- Addition: $\dfrac{p}{p \vee q}$
- Simplification: $\dfrac{p \wedge q}{q}$
- Disjunctive-syllogism: $\dfrac{p \vee q \quad \neg p}{q}$
- Hypothetical-syllogism: $\dfrac{p \rightarrow q \quad q \rightarrow r}{p \rightarrow r}$

Truth Tables for Connectives

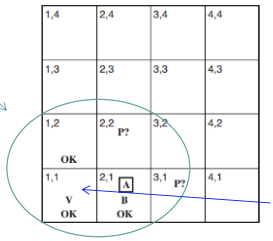
- **Summary:**

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Reduced Wumpus World

Wumpus World KB

Let's build the KB for the reduced Wumpus world.



agent mulai dari posisi (1,1): ruangan aman

- Let $P_{i,j}$ be true if there is a pit in $[i,j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i,j]$

$\neg P_{1,1}$

- "A square is breezy if and only if there is an adjacent pit"

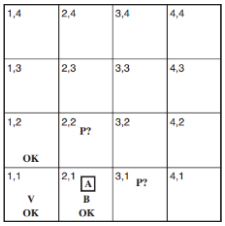
$B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$
$B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
$\neg B_{1,1}$
$B_{2,1}$

tidak ada *pit* di 1,1

gunakan simbol *propositional logic*

Wumpus World KB

Let's build the KB for the reduced Wumpus world.



- Let $P_{i,j}$ be true if there is a pit in $[i,j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i,j]$

R_1: $\neg P_{1,1}$

- "A square is breezy if and only if there is an adjacent pit"

R_2: $B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$
R_3: $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
R_4: $\neg B_{1,1}$
R_5: $B_{2,1}$

Questions: Based on this KB, is $KB \vDash P_{1,2}$? Is $KB \vDash P_{2,2}$?

For example we could wonder whether there is a pit in 1,2. In other words, does KB entails $P_{1,2}$, and does KB entails $P_{2,2}$?

Entailment and Inference

- **Semantics**: Determine entailment by **Model Checking**, that is enumerate all models and show that the sentence *a* must hold in all models.
$$KB \models \alpha$$
- **Syntax**: Determine entailment by **Theorem Proving**, that is apply rules of inference to KB to build a proof of *a without enumerating and checking all models*.
$$KB \vdash \alpha$$
- But how are entailment and inference related?

*) Jika *knowledge base* 'KB' *entails* 'α' maka semua interpretasi (baik menetapkan nilai 'benar' atau 'salah' ke variabel) yang mengevaluasi *knowledge* Anda ke True, juga mengevaluasi α ke True. $KB \models \alpha$.
*) Inferensi adalah prosedur untuk menurunkan kalimat baru 'α' dari 'KB' mengikuti beberapa algoritma. $KB \vdash \alpha$.

It's important to take a break here to express again entailments and inferences in terms of semantic and syntax.

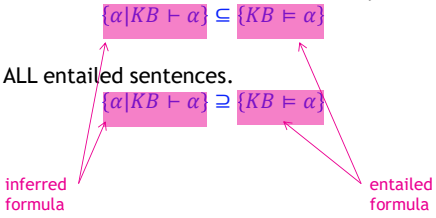
Semantics goes along with entailment. Entailment means determining, whether by model checking, a sentence alpha must be be true, or must hold, in all models in which KB is true. So we'll have to write it as KB models alpha, means that is alpha true whenever KB is true.

The syntax goes along with inference. And you know we like to determine entailment by what we call theorem proving. That is apply rule of inferences to the knowledge base to build a proof of alpha without having to explicitly enumerate all possible models or all truth tables for KB and for alpha. So we'll have to write this as either KB implies alpha, or KB infers alpha.

In the semantics we talk about truth tables, models. And in syntax we talk about inference rules, et cetera.

Soundness & Completeness

- We want an inference algorithm that is:
  1. **Sound**: does not infer false formulas, that is, derives only entailed sentences.
  2. **Complete**: derives ALL entailed sentences.



$\subseteq$: subset of or equal to

Dengan kata lain, agar algoritma inferensi menjadi *sound* berarti kita ingin semua rumus atau proposisi yang dapat dibuktikan menjadi benar adanya.

Dengan kata lain, agar algoritma inferensi menjadi *complete* berarti bahwa semua proposisi yang benar harus dapat dibuktikan menggunakan algoritma inferensi.

Validity & Satisfiability

- A sentence is valid (aka tautology) if it is true in all models,
  e.g., $True$, $p \vee \neg p, p \Rightarrow p, \big(p \wedge (p \Rightarrow q)\big) \Rightarrow q$
- Validity is connected to inference via the Deduction Theorem:
  $KB \vDash \alpha \; IFF \; (KB \Rightarrow \alpha)$ is valid
- A sentence is satisfiable if it is true in some model
  e.g., $p \vee q, r$
- A sentence is unsatisfiable if it is true in no models
  e.g., $p \wedge \neg p$
- Satisfiability is connected to inference via the following:

  $KB \vDash \alpha \; IFF \; (KB \wedge \neg \alpha)$ is unsatisfiable

  i.e., prove a by contradiction

Determining Entailment

how to determine entailments?

- Given a Knowledge Base (KB) (set of sentences in PL), given a query *a*, output whether KB entails α, noted: $KB \models \alpha$
- We will see two ways of doing proofs in PL:
  - **Model checking** enumerate the models (truth table enumeration, exponential).
  - **Application of inference rules** (proof checking/theorem proving): Syntactic derivations with rules like Modus Ponens (Backward chaining and forward chaining). A proof is a sequence of inference rule applications.

Model Checking and Inference

Model Checking

- Truth Table for inference
- Model: assignment T/F to every propositional symbol.
- Check that α is true in every model in which KB is true.

Here we are going to enumerate all possible models using truth tables.

Model Checking

- Truth Table for inference
- Model: assignment T/F to every propositional symbol.
- Check that α is true in every model in which KB is true.

a set of seven propositional symbols

a set of five rules

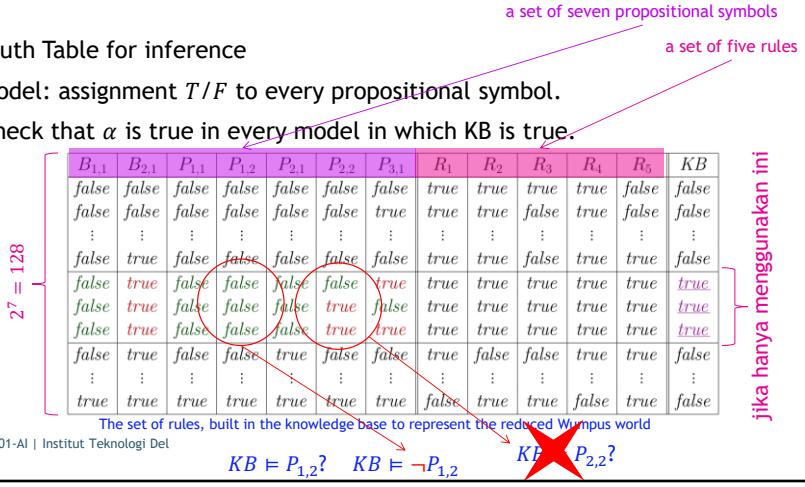
$2^7 = 128$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

The set of rules, built in the knowledge base to represent the reduced Wumpus world

Model Checking

- Truth Table for inference
- Model: assignment T/F to every propositional symbol.
- Check that α is true in every model in which KB is true.



Inference: Wumpus World

R_1: $\neg P_{1,1}$
R_2: $B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$
R_3: $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
R_4: $\neg B_{1,1}$
R_5: $B_{2,1}$

$KB \vDash P_{1,2}$? $KB \vDash P_{2,1}$?

R_2: $B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$

$\Leftrightarrow$

R_6: $(B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1} \Rightarrow B_{1,1})$

$\wedge$

$P_{1,2} \vee P_{2,1} \Rightarrow B_{1,1}$

Contrap.

$\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$

de Morgan's Law

R_4: $\neg B_{1,1}$

$\neg B_{1,1} \Rightarrow \neg P_{1,2} \wedge \neg P_{2,1}$

Modus Ponen

$\neg P_{1,2} \wedge \neg P_{2,1}$

$\wedge$

$\neg P_{1,2}$ $\neg P_{2,1}$

$KB \vDash \neg P_{1,2}$ $KB \vDash \neg P_{2,1}$

Now suppose we don't have model checking as a tool, we only can do inference using Modus Ponens and whatever equivalence propositional logic formula we found earlier.

As we can see, no semantics at all, just syntax.

Inference As A Search Problem

- **Initial state**: The initial KB
- **Actions**: all inference rules applied to all sentences that match the top of the inference rule
- **Results**: add the sentence in the bottom half of the inference rule
- **Goal**: a state containing the sentence we are trying to prove.

$KB \vDash P_{1,2}?$

You might have noticed something from the previous example, which is that inference sounds like search. On we're searching through what rules to apply, what inference to do to achieve some goal, which is to put entailments between the knowledge base and the proposition. So it's possible actually to cast the problem of inference as a search problem and use any of the algorithms we have seen in the previous weeks.

So…

Initial state: The initial KB

Actions: all inference rules applied to all sentences that match the top of the inference rule

Results: add the sentence in the bottom half of the inference rule

Goal: a state containing the sentence we are trying to prove.

Theorem Proving and Proof by Resolution

Theorem Proving

- Search for proofs is a more efficient way than enumerating models (We can ignore irrelevant information)
- Truth tables have an exponential number of models.
- The idea of inference is to repeat applying *inference rules* to the KB.
- Inference can be applied whenever suitable premises are found in the KB.
- Inference is sound. How about completeness?

Search for proofs is a more efficient way than enumerating models. Why? The main reason is that, first of all, truth tables are exponentials, and the number of propositions. And furthermore, when we do inference, we can simply ignore irrelevant information.

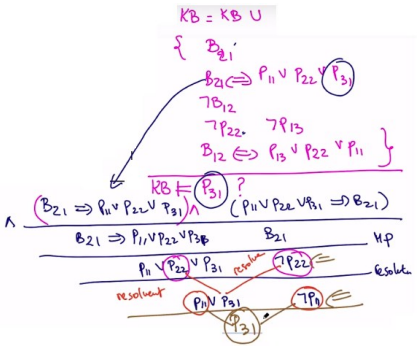
Theorem Proving

- Two ways to ensure completeness:
  - **Proof by resolution**: use powerful inference rules (resolution rule)
  - **Forward or Backward chaining**: use of modus ponens on a restricted form of propositions (Horn clauses)
- Resolution: ONE single inference rule
- Invented by Robinson, 1965
- Resolution + Search = complete inference algorithm.

Proof by Resolution

- Resolution & Wumpus world:

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1



Proof by Resolution

- **Unit resolution**:

$$\frac{l_1 \vee \cdots \vee l_k \qquad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

  where l_i and m are complementary literals.

- Example:

$$\frac{P_{1,3} \vee P_{2,2} \qquad \neg P_{2,2}}{P_{1,3}}$$

- We call a **clause** a disjunction of literals.
- Unit resolution: Clause + Literal = New clause.

Proof by Resolution

- **Resolution inference rule (for CNF)**:

$$\frac{l_1 \vee \cdots \vee l_k \qquad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where l_i and m_j are complementary literals.

- Resolution **applies only to clauses**
- **Fact**: Every sentence in PL is logically equivalent to a conjunction of clauses.
- **Conjunctive Normal Form (CNF)**: Conjunction of disjunction of literals:
  - Example: $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Resolution inference rule (for CNF): sound and complete for propositional logic

Conversion to CNF and Resolution Algorithm

Conversion to CNF

$$B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
$$\left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right) \wedge \left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right)$$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$
3. Move $\neg$ inwards using de Morgans rules and double-negation:
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge \left((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}\right)$$
4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

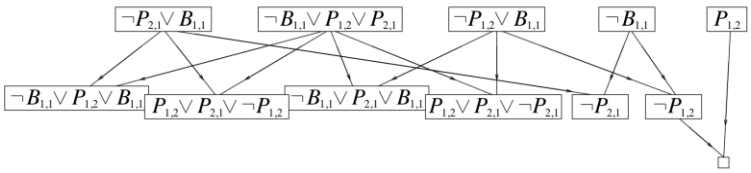
Resolution Algorithm

```
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 ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  loop do
      for each C_i, C_j in clauses do
          resolvents ← PL-Resolve(C_i, C_j)
          if resolvents contains the empty clause then return true
          new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution Example

$$KB = R_4 \wedge R_2 = (B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}) \wedge \neg B_{1,1}$$
$$\alpha = \neg P_{1,2}$$



Forward and Backward Chaining

Forward/Backward Chaining

- KB = conjunction of Horn clauses
- Horn clauses: logic proposition of the form: $p_1 \wedge \cdots \wedge p_n \Rightarrow q$
- Modus Ponens (for Horn Form): complete for Horn KBs

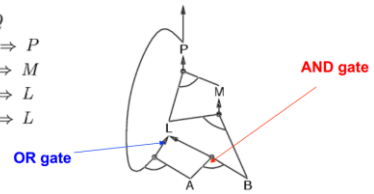
$$\frac{p_1, \cdots, p_n \qquad p_1 \wedge \cdots \wedge p_n \Rightarrow q}{q}$$

- Can be used with **forward chaining** or **backward chaining**.
- These algorithms are very natural and run in linear time

Forward Chaining

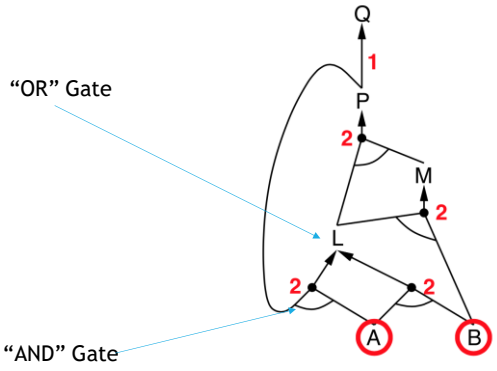
- **Idea**: Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found.
- This proves that $KB \Rightarrow Q$ is true in all possible worlds (i.e. trivial), and hence it proves entailment.

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



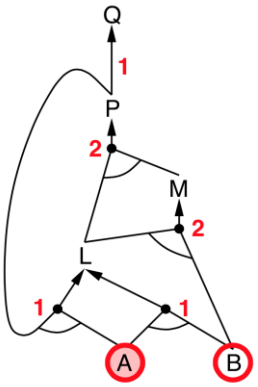
- Forward Chaining is sound and complete for Horn KB

Forward Chaining Example

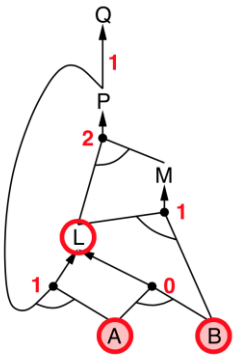


Note:
Perhatikan KB, L didapatkan baik dari rule ke-5 maupun rule-6, itulah sebabnya "OR" Gate digambarkan sebelum node L.

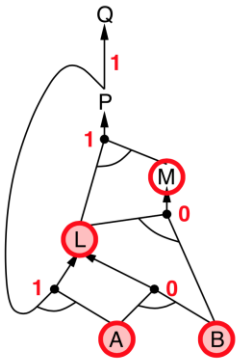
Forward Chaining Example



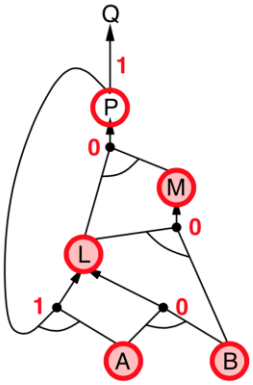
Forward Chaining Example



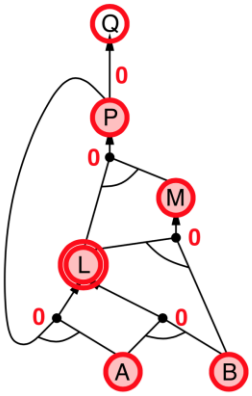
Forward Chaining Example



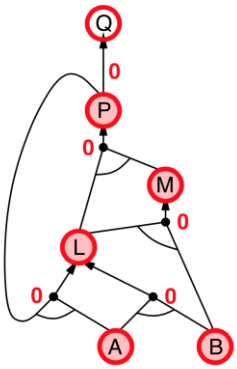
Forward Chaining Example



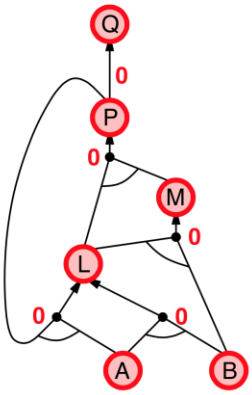
Forward Chaining Example



Forward Chaining Example



Forward Chaining Example

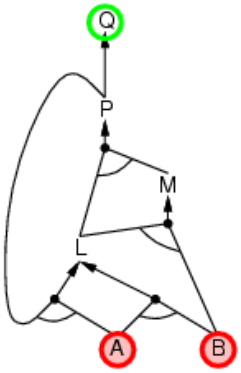


Backward Chaining

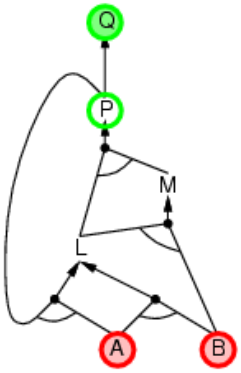
- Idea: Works backwards from the query q
- to prove q by Backward Chaining (BC):
  - Check if q is known already, or
  - Prove by BC all premises of some rule concluding q
  - Hence BC maintains a stack of sub-goals that need to be proved to get to q
- Avoid loops: check if new sub-goal is already on the goal stack
- Avoid repeated work: check if new sub-goal
  - has already been proved true, or
  - has already failed

Backward Chaining Example

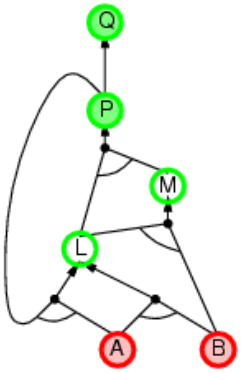
Note:
The idea is to go down in the graph until it reaches the facts here, A and B, and bring that information up.



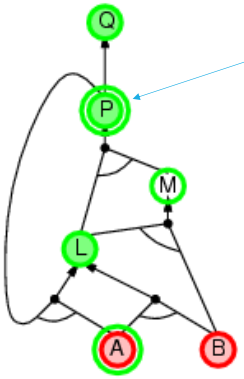
Backward Chaining Example



Backward Chaining Example

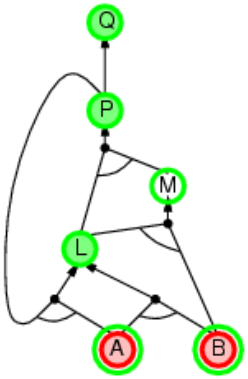


Backward Chaining Example



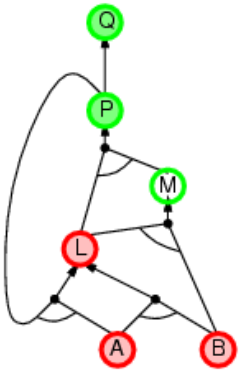
Note:
Kita perlu P untuk membuktikan L dan L untuk membuktikan P

Backward Chaining Example

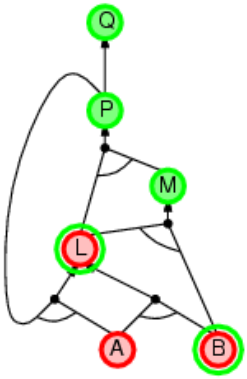


Note:
As soon as you can move forward, do so.

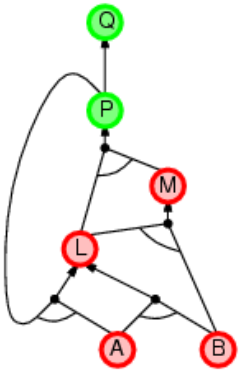
Backward Chaining Example



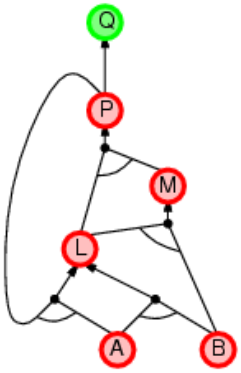
Backward Chaining Example



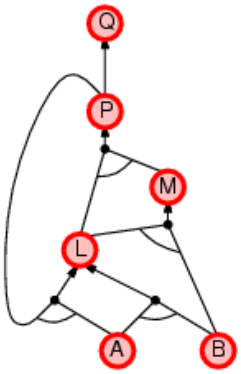
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



Forward vs. Backward

- Forward chaining is data-driven, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
- Backward chaining is goal-driven, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
  - Complexity of BC can be much less than linear in size of KB

Limits of Propositional Logic

- Limits of PL?
  - PL is not expressive enough to describe all the world around us. It can't express information about different object and the relation between objects.
  - PL is not compact. It can't express a fact for a set of objects without enumerating all of them which is sometimes impossible.
- Example: We have a vacuum cleaner (Roomba) to clean a 10×10 squares in the classroom. Use PL to express information about the squares.

Limits of Propositional Logic

- The proposition $square_1\_is\_clean$ expresses information about square1 being clean. How can one write this in a less heavy way?
- How can we express that all squares in the room are clean?

$$square_1\_is\_clean \wedge square_2\_is\_clean \wedge \ldots \wedge square_{100}\_is\_clean$$

- How can we express that some squares in the room are clean?

$$square_1\_is\_clean \vee square_2\_is\_clean \vee \ldots \vee square_{100}\_is\_clean$$

- How can we express that some squares have chairs on them?

$$square_1\_has\_chair \vee square_2\_has\_chair \vee \ldots \vee square_{100}\_has\_chair$$

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - **Syntax**: formal structure of sentences
  - **Semantics**: truth of sentences wrt models
  - **Entailment**: necessary truth of one sentence given another
  - **Inference**: deriving sentences from other sentences
  - **Soundness**: derivations produce only entailed sentences
  - **Completeness**: derivations can produce all entailed sentences

Summary

- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Forward, backward chaining are linear in time, complete for Horn clauses.
- Resolution is sound and complete for propositional logic.
- Propositional logic lacks expressive power.

References

- S. J. Russell and P. Borvig, Artificial Intelligence: A Modern Approach (4th Edition), Prentice Hall International, 2020.
- Kevin C. Klement, “Propositional Logic”, https://www.iep.utm.edu/prop-log/.
- AITopics: An official publication of the Association for the Advancement of Artificial Intelligence (AAAI), https://aitopics.org/search.

eof