Introduction to Knowledge-based Systems (WS2021) - Summary

Manuel Waibel

April 22, 2022

Contents

1 Predicate logic **4**
- 1.1 Closed formulas . . . 4
- 1.2 Model vs. Interpretation . . . 4
- 1.3 Entailment . . . 4
  - 1.3.1 Two types of entailment ($\models$) according to the lectures . . . 4
  - 1.3.2 Deduction theorem . . . 5
- 1.4 Equivalent replacement Lemma & Equivalent replacement theorem . . . 5
  - 1.4.1 Equivalent replacement Lemma . . . 5
  - 1.4.2 Equivalent replacement theorem . . . 5
- 1.5 Negation normal Form - NNF . . . 5
  - 1.5.1 Rules: . . . 5
- 1.6 Tableau Calculus for PL1 - TC1 . . . 5
  - 1.6.1 Rules . . . 6
  - 1.6.2 Properties . . . 6

2 Non-monotonic Reasoning **7**
- 2.1 Monotonic principle . . . 7
- 2.2 Deductive Closure $Cn(T)$ . . . 7
  - 2.2.1 Properties . . . 8
- 2.3 Closed-world assumption (CWA) . . . 8
  - 2.3.1 Computing T_{asm} . . . 8
  - 2.3.2 (In)Consistency . . . 9
- 2.4 Definite Horn clauses . . . 9
- 2.5 Generalizations of the CWA . . . 9

3 Non-monotonic Reasoning 2 **9**
3.1 Extensions . . . 10
3.1.1 Derive an extension . . . 10
3.2 Normal default . . . 11

4 Non-monotonic Reasoning 3 **11**

5 Answer-Set Programming 1 **11**
5.1 Rules in a logic program . . . 11
5.1.1 Types of rules . . . 12
5.2 Answer sets . . . 12
5.2.1 Gelfond-Lifschitz reduct . . . 12
5.3 Grounding a program . . . 14

6 Answer-Set Programming 2 **14**
6.1 Paradigms . . . 14
6.1.1 Guess-and-check Paradigm . . . 14
6.2 Safety . . . 15
6.3 Strong negation . . . 15
6.4 Commands . . . 15
6.4.1 Choice Rule/Subset . . . 15
6.4.2 #sum . . . 16

7 Probabilistic Reasoning **16**
7.1 Types of random variables . . . 16
7.2 Atomic event . . . 17
7.2.1 Properties . . . 17
7.3 Joint probability distribution . . . 17
7.4 Marginalisation . . . 17
7.5 Conditional probability . . . 18
7.6 Product rule . . . 18
7.7 Bayes rule . . . 19
7.8 Bayesian networks . . . 19
7.9 Reasonings . . . 20
7.9.1 Causal reasoning . . . 20
7.9.2 Diagnostic reasoning . . . 20
7.9.3 Intercausal reasoning . . . 20
7.10 D-separation . . . 21
7.10.1 Examples: . . . 21
7.11 Disadvantages of probabilistic reasoning . . . 23

8 Exam questions **23**

8.1 State the "equivalent replacement lemma" and the "equivalent replacement theorem". Using *first-order interpretation structures* and the "equivalent replacement lemma", show that the "equivalent replacement theorem" is valid. 23

8.1.1 Equivalent replacement Lemma 23

8.1.2 Equivalent replacement theorem 23

8.1.3 Proof 23

8.2 Prove or refute whether every default theory has an extension. 24

8.3 Define the notion of an *interpretation structure* of first-order predicate logic. Given a formula λ, explain the difference between an interpretation structure and a model for λ. Show or refute by means of interpretation structures: From $\psi \models \varphi \wedge \rho$ follows $\psi \models \rho$ for arbitrary closed formulas φ, ψ and rho. If you use additional theorems from the lecture you have to prove them. 24

8.4 Proofe or refute: $Cn(T_1) \cup Cn(T_2) \subseteq Cn(T_1 \cup T_2)$ 25

1 Predicate logic

1.1 Closed formulas

A formula φ is called closed, if it does not contain any free variables. That means the formula only contains constants, or every variable is i.e. bound by a quantifier ($\forall$ or $\exists$).

1.2 Model vs. Interpretation

An **Interpretation** ($=\Sigma$-structure) $\mathcal{I}$ is a **triple** $\langle \mathcal{U}, I, \alpha \rangle$.
- $\mathcal{U}$ is the domain of $\mathcal{I}$, i.e. a nonempty set of symbols
- I is the interpretation function
  - For constant symbols (0-ary) $c \in$ Func: $I(c) \in \mathcal{U}$
  - For n-ary function symbols $f \in$ Func: $I(f) : \mathcal{U}^n \mapsto \mathcal{U}$
  - For n-ary predicate symbols $p \in$ Pred: $I(p) \subseteq \mathcal{U}^n$
- α is a variable assignment

If $\mathcal{I}(\varphi) = 1$, (φ is a Σ-formula) then $\mathcal{I}$ is called a **model** of φ.
If φ has a model, then it is called **satisfiable** (if not all of them are it is also **refutable**).
If all $\mathcal{I}(\varphi)$ are models, then φ is a **tautology** ($=$ valid)
If $\mathcal{I}(\varphi)$ has no models, then φ is a **contradiction**

If a formula is a **tautology** ($=$ valid), it is also **satisfiable**!
If a formula is a **contradiction**, it is also **refutable**!

1.3 Entailment

$\alpha \models \beta$ iff every model where α is true, β is also true.

$\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$.

I.e. $(p = \text{True}) \models p \vee q$

An **empty conjunction** ($\wedge$) is **1** in **all** Σ-structures, i.e. is **equivalent to** $\top$.
An **empty disjunction** ($\vee$) is **0** in **all** Σ-structures, i.e. is **equivalent to** $\bot$.

1.3.1 Two types of entailment ($\models$) according to the lectures

1. Type:
$\models$ is related to an interpretation structure $\mathcal{I}$ and a formula φ.
$\mathcal{I} \models \varphi$ if the interpretation structure makes the formula φ true ($\mathcal{I}(\varphi) = 1$) and $\mathcal{I} \not\models \varphi$ if $\mathcal{I}(\varphi) = 0$.

2. Type:

$\models$ is related to a formula entailing another formula.
Let W be a set of closed formulas. Then $W \models \varphi$ iff $Mod(W) \subseteq Mod(\varphi)$ (every model of W is a model of φ).

1.3.2 Deduction theorem

$$W \cup \{\varphi\} \models \psi \text{ iff } W \models \varphi \implies \psi$$

1.4 Equivalent replacement Lemma & Equivalent replacement theorem

1.4.1 Equivalent replacement Lemma

Let $\langle \mathcal{U}, I, \alpha \rangle$be an interpretation structure with domain $\mathcal{U}$, interpretation function I and variable assignment α. If $\langle \mathcal{U}, I, \alpha \rangle \models \psi_1 \leftrightarrow \psi_2$, then $\langle \mathcal{U}, I, \alpha \rangle \models \varphi[\psi_1] \leftrightarrow \varphi[\psi_2]$

1.4.2 Equivalent replacement theorem

Let $\psi_1 \equiv \psi_2$. Then $\varphi[\psi_1] \equiv \varphi[\psi_2]$.

1.5 Negation normal Form - NNF

A formula is in NNF, if all negations $\neg$ are before ground terms. Usually $\rightarrow$ and $\leftrightarrow$ also get replaced.
$\neg(A \wedge B)$, is **not** in NNF.
$(\neg A \vee \neg B)$, **is** in NNF.

1.5.1 Rules:

De-Morgan Rules apply.

$$\begin{aligned}
(A \leftrightarrow B) &\equiv ((A \rightarrow B) \wedge (B \rightarrow A)) \\
(A \rightarrow B) &\equiv (\neg A \vee B) \\
(A \oplus B) &\equiv ((A \vee B) \wedge (\neg A \vee \neg B)) \\
\neg \forall x A &\equiv \exists x \neg A \\
\neg \exists x A &\equiv \forall x \neg A
\end{aligned}$$

1.6 Tableau Calculus for PL1 - TC1

1. Decompose a formula into their immediate subformulas
2. This leads to a decomposition tree
3. If a branch has a **clash** (i.e. p and $\neg p$) then that branch is **unsatisfiable**

4. If all branches have a **clash**: input formula is **unsatisfiable**
5. If the decomposition process stops and there is some branch **without a clash**, then the branch **provides a model**

If one has to show a **logical consequence**, following steps are needed:

1. Construct $\neg\varphi$ of the formula to show it is a consequence and bring in NNF
2. In the Tableau, insert the other statements
3. Decompose the formulas according to the Rules
4. If the formula is unsatisfiable (has clashes in every leaf) then the formula is a logical consequence of the knowledge base

1.6.1 Rules

A conjunction $\varphi \wedge \psi$ is written under each other.

$\varphi \wedge \psi \longrightarrow \begin{array}{l} \varphi \\ \psi \end{array}$

A disjunction $\varphi \vee \psi$ is written as a split.

$\varphi \vee \psi \longrightarrow \varphi \mid \psi$

A universal quantifier $\forall$ is replaced by a variable that is already used in the tableau.

$\forall x\varphi \longrightarrow \varphi\{x \leftarrow t\}$

A existential quantifier $\exists$ is replaced by a new Skolem constant

$\exists x\varphi \longrightarrow \varphi\{x \leftarrow a\}$

Existential quantifiers should always be replaced before universal quantifiers if possible!

1.6.2 Properties

Soundness: If there is a closed TC1-tableau for a first-order formula in NNF, then the formula is unsatisfiable.
Completeness: If a first-order formula in NNF is unsatisfiable, then there is a closed TC1-tableau for this formula.
If a formula is **satisfiable** then the TC1 **does not terminate in general**!
If a formula is **unsatisfiable** then the TC1 **does terminate**!

2 Non-monotonic Reasoning

2.1 Monotonic principle

$$\text{If } S \models A \text{ and } S \subseteq S', \text{ then } S' \models A$$

In monotonic reasoning a contrary fact to another does not change the outcome (see above subset definition).
Example:

$$\forall x Bird(x) \implies CanFly(x)$$
$$Bird(Tweety)$$
$$\rightarrow CanFly(Tweety) \quad \checkmark$$

Later adding the facts, that Tweety is a penguin and penguins cannot fly, the fact still remains valid!

Commonsense reasoning violates the monotonic principle. Current information is considered the only relevant one for a particular problem. Therefore the above reasoning is false in non-monotonic reasoning.
So it is possible by adding different facts, that a conclusion can no longer be derived (e.g. $W \subset W'$ but $E \not\subseteq E'$, since $Flies(Tweety) \in E$ but $Flies(Tweety) \notin E'$)

2.2 Deductive Closure $Cn(T)$

A formula T is deductivly closed, if it contains every formula φ that can be logically deducted from T.
Formally:

$$Cn(T) = \{\varphi \mid T \models \varphi \text{ and } \varphi \text{ is closed}\}$$

The deductive closure of a theory $Cn(T)$ is **always an <u>infinite</u> set**! Because it contains all trivial formulas like $\top \vee \top$, $\top \vee \top \vee \top \vee \ldots$, etc. and can therefore be expanded infinitely.

2.2.1 Properties

$T \subseteq Cn(T)$
$Cn(T) = Cn(Cn(T))$
$T \subseteq T' \implies Cn(T) \subseteq Cn(T')$
$Cn(\emptyset) \neq \emptyset$
$Cn(T)$ is the set of all formulas iff T is inconsistent
If $\varphi \in Cn(T)$, then $Cn(T \cup \{\varphi\}) = Cn(T)$

2.3 Closed-world assumption (CWA)

In the closed-world assumption, the given knowledge base is considered to be the only true one. Any other information, that is not in this closed-world is considered to be false.
The CWA completes the knowledge base by adding the missing (negative) facts (a theory T is complete iff, for each ground atom P, $P \in T$ or $\neg P \in T$ holds).

$$T_{asm} = \{\neg P \mid P \text{ ground atom}, T \not\models P\},$$
$$CWA(T) = \{\varphi \mid T \cup T_{asm} \models \varphi, \varphi \text{ closed}\} = Cn(T \cup T_{asm})$$

If T is inconsistent (e.g. contains $P(a)$ and $\neg P(a)$), then T_{asm} is empty!
Since $CWA(T) = Cn(T \cup T_{asm})$ the $CWA(T)$ is always infinite (not finite), because the $Cn(T)$ is infinite (see 2.2 Deductive Closure $Cn(T)$).

2.3.1 Computing T_{asm}

1. Extract every ground atom.
2. Check which of the ground atoms are in T (or can be deducted by it)
3. T_{asm} = negation of the ground atoms, which are not in T

If T is inconsistent (e.g. contains $P(a)$ and $\neg P(a)$), then T_{asm} is empty!
Example:

$$T = \{\forall x(p(x) \vee q(x)), p(a), q(b)\}$$

After step 1. & 2.:

	$p(a)$	$p(b)$	$q(a)$	$q(b)$
$T \models$	✓	×	×	✓

After 3.: $T_{asm} = \{\neg p(b), \neg q(a)\}$

2.3.2 (In)Consistency

Let T be a consistent theory. Then:
$CWA(T)$ is inconsistent iff there are ground atoms $A_1, ..., A_n$ such that $T \models A_1 \vee ... \vee A_n$, but $T \not\models A_i$, for all $i = 1, .., n$.
Example:

$$T = \{\forall x(p(x) \vee q(x)), p(a), q(b)\}$$
$$\rightarrow \text{ is consistent}$$

2.4 Definite Horn clauses

A definite Horn clause is a clause having **precisely one** positive literal.
Example:

$\neg p \vee \neg q \vee s$	is a definite Horn clause
$\neg p \vee \neg q$	is not a definite Horn clause

If a CWA only consists of Horn clauses, it is **consistent**.

2.5 Generalizations of the CWA

CWA relative to a predicate symbol (PS) e.g. q

Denoted as: $CWA^{\mathbf{q}}(T)$
So only concern about atoms using q

Formally:

$$T^q_{asm} = \{\neg P \mid P \text{ ground atom with PS } q, T \not\models P\}$$
$$CWA^q(T) = \{\varphi \mid T \cup T^q_{asm} \models \varphi, \varphi \text{ closed}\} = Cn(T \cup T^q_{asm})$$

Example $CWA^p(T)$:

$$T = \{\forall x(q(x) \implies p(x)), q(a), r(b) \vee p(b)\}$$
$\rightarrow \neg p(b) \in CWA^p(T)$, because $p(b)$ cannot be derived from T (is in T_{asm})
$\rightarrow \neg q(b) \in CWA^p(T)$, because $\neg p(b)$ to satisfy first expression
$\rightarrow r(b) \in CWA^p(T)$, because of $\neg p(b)$ to satisfy third expression

3 Non-monotonic Reasoning 2

A **default theory** is denoted as $T = (W, \Delta)$ where:

W	is the set of known facts (closed first-order formulas)
Δ	are the "defaults" or "default rules"

The basic idea of default reasoning is to apply the defaults in Δ to the facts in W derive plausible inferences.

Defaults are denoted like:

$$\frac{Prerequisites : Justification_1, ..., Justification_n}{Conclusion}$$

3.1 Extensions

An extension E of $T = (W, \Delta)$ tries to extend the knowledge base by applying the defaults to the extended knowledge base.

3.1.1 Derive an extension

1. Replace all existence-quantifier variables $(\exists x)$ in W with new scolem constants and all variables in Δ with those new constants. This step means building the closure of a default theory.

2. List all possible extension candidates by combining W with all possible conclusion of the defaults. This means $Cn(W \cup \{...\})$

3. Determine if E_i is inconsistent, by looking at the facts of W

   If inconsistent $\longrightarrow$ extension includes all formulas and can be ignored

   If consistent $\longrightarrow$ continue for this candidate with next steps

4. Determine Δ_{E_i} by looking at the negated(!) justifications of the defaults. Δ_{E_i} then consists of all *prerequisites/conclusion* pairs if the negated(!) justifications cannot be derived from $Cn(W \cup \{...\})$.

5. Determine $\Gamma_T(E_i)$ by applying the defaults of Δ_{E_i} to the facts of the extension and build the Cn of the set of facts (include inconsistencies). Also use W to derive necessary prerequisites or justifications!

6. Finally look if the set of facts from $\Gamma_T(E_i)$ match any other sets from the extensions.

   If so $\longrightarrow$ no extension

   If the set matches it's own extension $\longrightarrow$ Extension!

Example (12.01.2022 Exam):

(Step 1. and 2. were already done for us)

$$\Delta = \left\{ \frac{P(x) : Q(x)}{Q(x)}, \frac{Q(x) : P(x), R(x)}{P(x)}, \frac{\top : \neg P(x), \neg R(x)}{P(x) \wedge Q(x)} \right\}$$

$W_1 = \{Q(a), P(a), P(b), Q(b)\}$ $W_2 = \{\neg P(a), R(a), Q(a)\}$ $W_3 = \{R(a)\}$
$E_1 = Cn(W_1)$ $E_2 = Cn(W_2)$ $E_3 = Cn(W_3)$

$\Delta^{E_1} = \{P(a)/Q(a), P(b)/Q(b), Q(a)/P(a), Q(b)/P(b)\}$

In the first default is in Δ^{E_1}, because the negation of the justification ($\neg Q(a)$) cannot be derived from W_1 (or E_1). Repeat this for the second default and the constant b. The last default is not contained in Δ^{E_1}, because $\neg\neg P(a) = P(a)$ can be derived from W_1.

$\Delta^{E_2} = \{P(a)/Q(a)\}$

$\Delta^{E_3} = \{P(a)/Q(a), Q(a)/P(a)\}$

3.2 Normal default

A default is called **normal** if it has the form

$$\frac{\varphi : \psi}{\psi}$$

A default theory only consisting of **normal defaults** is called **normal** and **always has an extension**.

4 Non-monotonic Reasoning 3

5 Answer-Set Programming 1

5.1 Rules in a logic program

A rule r is denoted as:

$$\underbrace{a_1 \vee ... \vee a_m}_{\text{head}} :- \underbrace{b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_n}_{\text{body}}$$

5.1.1 Types of rules

Fact	If there exists a head and the body is empty. `P(x).`
Basic	No default negation ("*not*") and there is no empty head. `P(x) :- S(x).`
Non-disjunctive	If no disjunction ($\vee$) in head. `P(x) :- not S(x), T(x).`
Normal	If it is non-disjunctive and contains no strong negation $\neg$. `P(x) :- not S(x), T(x).`
Horn	If it is normal and basic. `P(x) :- S(x), T(x).`
Ground	If it has no variables (only constants). `P(a) :- not S(a), T(a).` (note lower-case `a` is a constant)

5.2 Answer sets

An answer-set of a program P is defined in terms of the **reduct** or **Gelfond-Lifschitz reduct** of a program.

An interpretation M is an **answer-set** of a ground program P iff it is a **minimal model** of P^M (reduct), i.e. there is no $N \subset M$ which is also a model of P^M.

M is a classical model of a program P, iff it contains all literals where:
Whenever the body of the rule is true, then the head must also be true.
Note, that M can be a classical model of P, but not an answer-set of P.

If an answer-set is a consistent set of ground literals, then it is called an *interpretation*.

5.2.1 Gelfond-Lifschitz reduct

Is used to calculate answer-sets, or to prove/show that a given set is a answer-set.

Algorithm:

Take a candidate set and then...

1. delete all rules, where negative body is in conflict with the candidate
2. delete from remaining rules all negative literals.

   E.g. $R(a,a) \leftarrow P(a),\ not\ S(a).$ becomes: $R(a,a) \leftarrow P(a).$

3. The set of any facts or rules remaining, is the reduct of $\mathcal{P}$

Example:

$$\mathcal{P} := \left\{ \begin{array}{r} a \vee c \leftarrow not\ c. \\ a \leftarrow not\ c. \\ b \leftarrow not\ c. \end{array} \right\}.$$

The answer-set is $S_1 = \{a, b\}$ (if $not\ c$).
Now to test if S_1 is a answer-set we calculate the Gelfond-Lifschitz reduct as follows:

1. Delete all rules, where negative body is in conflict with the candidate:
Since c is not contained in the answer-set no rule ($not\ c$) stands in conflict, so we keep all rules.

2. Delete from remaining rules all negative literals.
After this step the program $\mathcal{P}$ looks like this:

$$\mathcal{P}^{S_1} := \left\{ \begin{array}{r} a \vee c \leftarrow . \\ a \leftarrow . \\ b \leftarrow . \end{array} \right\}.$$

Which is the same as:

$$\mathcal{P}^{S_1} := \left\{ \begin{array}{r} a \vee c. \\ a. \\ b. \end{array} \right\}.$$

Is S_1 a minimal model of $\mathcal{P}^{S_1} = \{a \vee c., a., b.\}$?
Yes, because every model must make a and b true and either a or c true. With our answer-set S_1 this is the case. Also no other sub- or superset of this answer-set is possible (no: $\{a\}$, $\{b\}$, $\{a, b, c\}$, ... or $\{\}$).

For the (rather obvious) case that if c is true there is no answer-set for that solution can also be shown by the reduct.
Consider the answer-set $S_2 = \{c\}$ to make c true (rules exclude other literals from the answer-set).
After step **1.** of the algorithm no rules are left in $\mathcal{P}^{S_2}$, because every rule body ($not\ c$) is in conflict with the c of the answer-set. This means, that $\mathcal{P}^{S_2} = \{\}$ is the result (because the minimal reduct of $\{\}$ is $\{\}$).
Since $S_2 \neq \{\}$, it holds that S_2 is no answer-set (and therefore every superset of $\{c\}$).
Therefore S_1 is the only answer-set of $\mathcal{P}$.

5.3 Grounding a program

To ground a program, all variables are replaced by constants.

Example:

$$\mathcal{P} := \left\{ \begin{array}{l} C(r).\ P(t). \\ F(X) \leftarrow B(X),\ not\ \neg F(X). \\ \neg F(X) \leftarrow P(X). \\ B(X) \leftarrow P(X). \\ B(X) \leftarrow C(X). \end{array} \right\}.$$

The grounding of $\mathcal{P}$ ($grnd(\mathcal{P})$) is:

$F(r) \leftarrow B(r),\ not\ \neg F(r).$	Replace the variable X by the constant r
$F(t) \leftarrow B(t),\ not\ \neg F(t).$	Replace the variable X by the constant t
$\neg F(r) \leftarrow P(r).$	Replace the variable X by the constant r
$\neg F(t) \leftarrow P(t).$	Replace the variable X by the constant t
$B(r) \leftarrow P(r).$	Replace the variable X by the constant r
$B(t) \leftarrow P(t).$	Replace the variable X by the constant t
$B(r) \leftarrow C(r).$	Replace the variable X by the constant r
$B(t) \leftarrow C(t).$	Replace the variable X by the constant t

6 Answer-Set Programming 2

6.1 Paradigms

6.1.1 Guess-and-check Paradigm

Guess defines the search space.
Check removes unwanted branches.

Also known as "generate-define-test" methodology.
Generate a search space
Define auxiliary predicates
Test whether all conditions are satisfied.

Approach:

1. Guess a solution candidate

   Using cyclic negations, disjunctions, ...

2. Check admissibility of candidate

   Using constraints and possibly auxilary predicates to remove candidates that fail the required conditions

6.2 Safety

Variables in rules must be bound to be instantiated (replaced by a constant).
A Variable V is bound of it occurs in the body or outside an arithmetic term in:

- a classical (positive) atom
- a comparison predicate as $V = ...$ (cycle-free)
- an aggregate $V = \#agg...$ (cycle-free)

A rule is save if all its variables are bound.

6.3 Strong negation

Contrary to the *neg* keyword, which indicates, that a literal cannot be derived, the strong negation indicates the logical counterpart.

It is not possible that both `p` and `-p` are true.
It is possible that both `p` and `-p` are false in the same answer-set. (e.g. if `p` is not derivable).

Example:

```
a.
:- p, a.
:- -p, a.
```

Has the answer-set $\{a\}$, because `p` is not derivable and therefore `p` and `-p` are both false.

6.4 Commands

6.4.1 Choice Rule/Subset

```
{a; b} :- c.
```

If `c` is holds, any subset of `a` and `b` may be true (this will generate any subset of `a` and `b`).
It also allows boundaries to be set, e.g.:

```
2 <= {a; b; c; d} <= 3 :- p.
```

If `p` holds, then between 2 and 3 of `a`, `b`, `c` or `d` are true.

6.4.2 #sum

Sum will only sum up **unique** numbers unless "specified".
Example:

```
p(1, 3).
p(3, 3).
p(1, 1).
p(1, 4).
p(7, 2).

q(4, 5).
q(8, 5).
q(9, 1).
q(0, 7).


s(X) :- X = #sum{A : p(A, _);C : q(_, C)}.
```

Will produce the output `16`, because the unique numbers summed up are $(1, 3, \not{1}, \not{1}, 7, 5, \not{5}, \not{1}, \not{7} = 16)$
Using `s(X) :- X = #sum{A, B : p(A, B); C, D : q(D, C)}.` will look at the unique pairs of numbers (result is `31`).

7 Probabilistic Reasoning

7.1 Types of random variables

- **Boolean random variable:** has the domain $\{true, false\}$.

  Example: Variable *Dead* can either be $true$ or $false$.

  True variables often are denoted by lower case names, for example if $Dead = true$, one could write it as *dead*. If $Dead = false$ then it can be denoted as $\neg dead$

- **Discrete random variables:** can take values from a countable domain.

  The values must be mutually exclusive and exhaustive (must leave no logical gaps)!

  Example: The variable $Weather$ can be $\{sunny, rainy, cloudy, snow\}$

- **Continuous random variables:** can take values from real numbers (e.g. $X = 4.02$).

  With continuous of random variables inequalities can also be used (e.g. $X \leq 4.02$).

7.2 Atomic event

A atomic event is an assignment of values to all the random variables used. This means, that every random variable gets assigned a value of the domain.

Example:
If the world consists of the Boolean variables *Dead* and *Alive* then all atomic events are:
$Dead = true \wedge Alive = true$
$Dead = true \wedge Alive = false$
$Dead = false \wedge Alive = true$
$Dead = false \wedge Alive = false$

7.2.1 Properties

- Atomic events are **mutually exclusive** (only one can be true)
- **At least one** has to be true

  This means, that the disjunction of all atomic events is logically equivalent to $\top$

7.3 Joint probability distribution

Let $V_1, ..., V_n$ be some random variables and Ω the set of all atomic events over $V_1, ..., V_n$.

A **joint probability distribution** (JPD) of $V_1, ..., V_n$ is a function P, that maps a number $P(\omega)$ to each atomic event in $\omega = (V_1 = v_1, ..., V_n = v_n)$ (variable V_i has the value v_i) such that:

- $0 \leq P(\omega) \leq 1$
- $\sum_{\omega \in \Omega} P(\omega) = 1$

$P(V_1, ..., V_n)$ refers to the JPD P of V_1

7.4 Marginalisation

Marginalisation means, calculating a (general) probability with a set of (specific) probabilities.
For Example calculating $P(Y = 1)$ if given:
$P(X = 0, Y = 0) = 0.5$
$P(X = 0, Y = 1) = 0.1$
$P(X = 1, Y = 0) = 0.1$
$P(X = 1, Y = 1) = 0.3$

$P(Y = 1)$ is then calculated as: $P(Y = 1) = P(X = 0, Y = 1) + P(X = 1, Y =$

$1) = 0.1 + 0.3 = 0.4$

Generally:
Below notation means, that we sum over all probabilities, where $V_i = v_i$. So we sum up all probabilities, where V_i has the specific value v_i.

$$P(V_i = v_i) = \sum_{V_i = v_i} P(V_1, ..., V_n)$$

Or

$$P(V_{i_1} = v_{i_1}, ..., V_{i_k} = v_{i_k}) = \sum_{V_{i_1} = v_{i_1}, ..., V_{i_k} = v_{i_k}} P(V_1, ..., V_n)$$

7.5 Conditional probability

Conditional probability $P(V_i = v_i \mid V_j = v_j)$: gives the probability that $V_i = v_i$ by given evidence $V_j = v_j$.
"Gives the probability of $V_i = V_i$ if all the agent knows is $V_j = v_j$."
$P(A \mid B) \rightarrow$ "What is the probability of A happening, given that B already happened?" (so the probability that A happens after B)

$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

(If A and B independent: $P(A, B) = P(A \cap B) = P(A) \cdot P(B)$)

Example with multiple Boolean variables:

$$P(b, \neg t \mid \neg a, w) = \frac{P(b, \neg t, \neg a, w)}{P(\neg a, w)}$$

7.6 Product rule

$P(A) = P(A)$
$P(A \cap B) = P(A \mid B) \cdot P(B)$
$P(A \cap B \cap C) = P(A \mid B, C) \cdot P(B \mid C) \cdot P(C)$
$\vdots$

This results to the product rule (factors are independent of order):

$$P(V_1, ..., V_n) = \prod_{i=1}^{n} P(V_i \mid V_{i+1}, ..., V_n)$$

7.7 Bayes rule

Bayes rule:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

In words: The probability of A happening given B ("$P(A \mid B)$") is the same as, that B is happening given A times the probability of A, devided by B.

So the events A and B can be swapped.
Consider:

$$P(A \mid B) = P(A \mid B) \cdot P(B)$$
$$P(A \mid B) = P(B \mid A) \cdot P(A)$$

$$P(A \mid B) = \underline{\underline{\frac{P(B \mid A) \cdot P(A)}{P(B)}}}$$

7.8 Bayesian networks

A Bayesian network is a directed graph representing the JPD and taking **independence of random variables** into account.

- Each **node** of the network is a **random variable**.
- Each **edge** represents a **dependency** between the nodes.

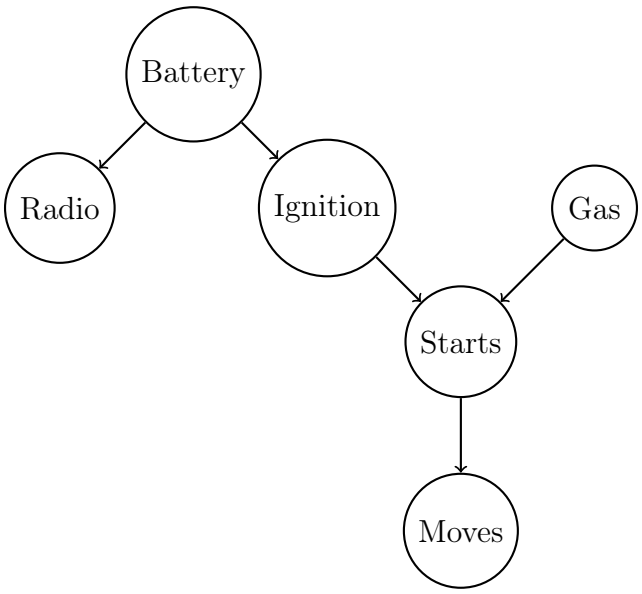
  A child node is dependent on its direct parent.

  B is a child of A, if $A \rightarrow B$.
- It is a directed acyclic graph (DAG).
- The topology of the network specifies the conditional independence between nodes/variables.

  Each node is conditionally independent of its non-descendants, given its parents as evidence.
- The network provides full information about the JPD.

  $P(V_1, ..., V_n) = \prod_{i=1}^{n} P(V_i \mid Parents(V_i))$

Example:



Here the node "Starts" **is** dependent on "Ignition" and "Gas", but **not** dependent on **any other node**.
Therefore $P(Starts \mid Battery, Radio, Ignition, Gas) = P(Starts \mid Ignition, Gas)$.
But $P(Ignition \mid Battery, Starts) \neq P(Ignition \mid Battery)$.

7.9 Reasonings

7.9.1 Causal reasoning

Goes from **causes to effects**.
Example: $P(Starts \mid Ignition) \rightarrow$ the probability of the engine starting, given the event of the ignition.

7.9.2 Diagnostic reasoning

Goes from **effects to causes** (think doctor on health related issues).
Example: $P(\neg Ignition \mid \neg Starts) \rightarrow$ the probability of the ignition being broken, if the engine won't start.
Can be reduced to causal reasoning by Bayes' rule.

7.9.3 Intercausal reasoning

Is between **causes of a common effect**.
Example: $P(\neg Ignition \mid \neg Starts, \neg Gas) \rightarrow$ the probability of the ignition not working, given that the engine won't start and not having gas.
In this scenario $\neg Ignition$ becomes less likely, because we already have an explanation, why the engine won't start, namely that it has no gas. This is called "explaining away".

7.10 D-separation

Video to d-seperation: `https://youtu.be/yIwTzdwVz0Q`

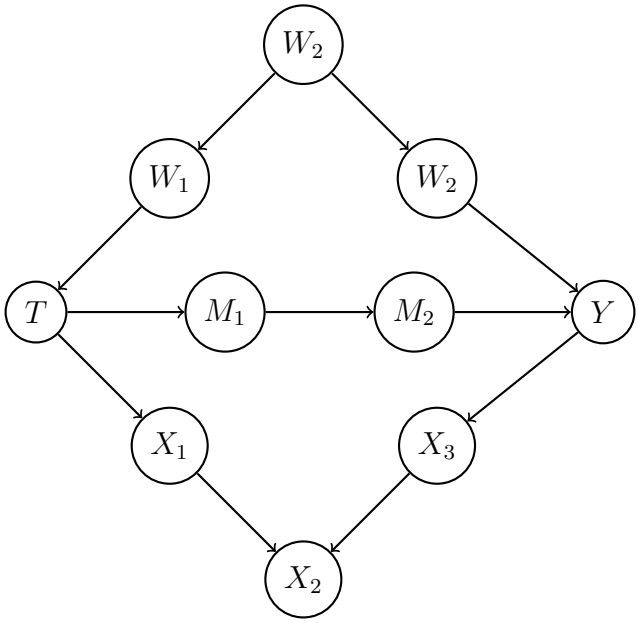
D-separation is used to describe relationships between other nodes, if a certain node is observed (e.g. in the graph above: if we observe the amount of gas we have, can we say something about the relationships of other nodes?).

Given a set of observed nodes E, a path between two nodes is **blocked** if along the path there is a...

1. **chain**, which a node from E is in ($... \rightarrow W \rightarrow ...$ and $W \in E$)
2. **fork**, which a node from E is in ($... \leftarrow W \rightarrow ...$ and $W \in E$)
3. **collider** W ($... \rightarrow W \leftarrow ...$) which is not conditioned on ($W \notin E$) and none of its descendants are conditioned on ($de(W) \not\subseteq E$).

Blocked paths between two nodes mean, that they are conditionally independent.

7.10.1 Examples:



Example 1:
Question: Are T and Y d-separated given $E = \{\}$?
Answer: No, because there is a path from T to Y using M_1 and M_2 (or the W variables, but not the X variables because X_2 is a collider)!

Example 2:
Question: Are T and Y d-separated given $E = \{M_1\}$?

Answer: No, because there is a path from T to Y using W_1, W_2 and W_3 (but not the X variables because X_2 is a collider)!

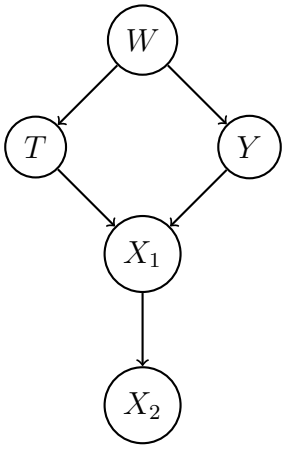
Example 3:
Question: Are T and Y d-separated given $E = \{M_1, W_2\}$?
Answer: Yes, because all paths are blocked. M_1 builds a chain, W_2 builds a fork and X_2 builds a collider!

Example 4:
Question: Are T and Y d-separated given $E = \{M_1, W_3\}$?
Answer: Yes, because M_1 and W_3 are both building a chain and X_2 is a collider!

Example 5:
Question: Are T and Y d-separated given $E = \{M_1, W_2, W_3\}$?
Answer: Yes, because M_1, W_2 and W_3 are both building a chain and X_2 is a collider!

Example 6:
Question: Are T and Y d-separated given $E = \{M_1, X_2, W_2, W_3\}$?
Answer: No, because the path through the X variables is now unblocked, because the previous collider X_2 is now not a collider anymore!

Example 7:
Question: Are T and Y d-separated given $E = \{M_1, X_1, X_2, W_2, W_3\}$?
Answer: Yes, because X_1, W_2 and W_3 are all contained in a chain!



Example 8:
Question: Are T and Y d-separated given $E = \{W, X_2\}$?
Answer: No, because conditioning on X_2, X_1 is not a collider and therefore there is a path through X_1 (path through W is blocked)!

Example 9:
Question: Are T and Y d-separated given $E = \{W\}$?
Answer: Yes, because W builds a fork and X_1 is a collider!

7.11 Disadvantages of probabilistic reasoning

- Generally large datasets are required to determine the needed conditional probabilities.
- Difficult to verify whether independence between two variables is actually the case.
- In contrast to nonmomotonic logics, probabilistic reasoning does not allow to represent non-knowledge. Each atomic event must have a assigned probability.
- Inconsistent information is not detected and is propagated further on.

8 Exam questions

8.1 State the "equivalent replacement lemma" and the "equivalent replacement theorem". Using *first-order interpretation structures* and the "equivalent replacement lemma", show that the "equivalent replacement theorem" is valid.

8.1.1 Equivalent replacement Lemma

Let $\langle \mathcal{U}, I, \alpha \rangle$be an interpretation structure with domain $\mathcal{U}$, interpretation function I and variable assignment α. If $\langle \mathcal{U}, I, \alpha \rangle \models \psi_1 \leftrightarrow \psi_2$, then $\langle \mathcal{U}, I, \alpha \rangle \models \varphi[\psi_1] \leftrightarrow \varphi[\psi_2]$

8.1.2 Equivalent replacement theorem

Let $\psi_1 \equiv \psi_2$. Then $\varphi[\psi_1] \equiv \varphi[\psi_2]$.

8.1.3 Proof

If $\psi_1 \equiv \psi_2$, then $\psi_1 \leftrightarrow \psi_2$ holds for every interpretation I because:

If $\psi_1 \equiv \psi_2$ then every model of ψ_1 is a model of ψ_2 ($Mod(\psi_1) = Mod(\psi_2)$). If we take a look at $\psi_1 \leftrightarrow \psi_2$ we can perform a case split for a arbitrary interpretation I:

1. $I \models \psi_1$, then also $I \models \psi_2$, so $I \models \psi_1 \leftrightarrow \psi_2$
2. $I \not\models \psi_1$, then also $I \not\models \psi_2$, so $I \models \psi_1 \leftrightarrow \psi_2$

According to the lemma, it holds $I \models \varphi[\psi_1] \leftrightarrow \varphi[\psi_2]$.
Because we chose an arbitrary interpretation, $\varphi[\psi_1] \leftrightarrow \varphi[\psi_2]$ is a tautology and therefore it holds: $\varphi[\psi_1] \equiv \varphi[\psi_2]$. $\square$

8.2 Prove or refute whether every default theory has an extension.

This is **false**. We construct a counterexample:

$T = \langle \emptyset, \left\{ \frac{\top : \neg P(a)}{P(a)} \right\} \rangle$

Extension candidates:
$E_1 = Cn(\emptyset)$
$\Delta^{E_1} = \{\top / P(a)\}$
$\Gamma_T(E_1) = Cn(P(a)) = E_2 \neq E_1$

$E_2 = Cn(P(a))$
$\Delta^{E_1} = \{\}$
$\Gamma_T(E_2) = Cn(\emptyset) = E_1 \neq E2$

T has no extensions. □

8.3 Define the notion of an *interpretation structure* of first-order predicate logic. Given a formula λ, explain the difference between an interpretation structure and a model for λ. Show or refute by means of interpretation structures: From $\psi \models \varphi \wedge \rho$ follows $\psi \models \rho$ for arbitrary closed formulas φ, ψ and rho. If you use additional theorems from the lecture you have to prove them.

(Not a confirmed, proven solution)

An interpretation structure $\mathcal{I}$ is a triple $\langle \mathcal{U}, I, \alpha \rangle$, where $\mathcal{U}$ is the domain, I is the interpretation function and α are variable assignments.

If $\mathcal{I}(\lambda) = 1$, then $\mathcal{I}$ is called a **model** of λ.

Proof/Counterexample: $\psi \models \varphi \wedge \rho \implies \psi \models \rho$, for arbitrary formulas ψ, φ and ρ.

Let's say there is an interpretation structure $\mathcal{I}$, for which $\mathcal{I} \models \varphi \wedge \rho$. For the entailment to hold every model of $\mathcal{I}$ must be a model of $\varphi \wedge \rho$ (by definition of the entailment). So $\mathcal{I}$ is a model of if both $\varphi \wedge \rho$ are true.
If $\mathcal{I} \models \varphi \wedge \rho$, then $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \rho$ and therefore the implication is correct. □

8.4 Proofe or refute: $Cn(T_1) \cup Cn(T_2) \subseteq Cn(T_1 \cup T_2)$

(Not a confirmed, proven solution)

The deductive closure Cn of a theory T consists of all formulas, which can be logically deducted from T.
If a theory T contains inconsistencies, the deductive closure is the set of all formulas.

For the proof we perform a case split if T_1 and T_2 contain inconsistensies, towards each other (e.g. $T_1 = \{P(a)\}$ and $T_2 = \{\neg P(a)\}$):

T_1 and T_2 make each other inconsistent:
Then $Cn(T_1) \cup Cn(T_2)$ is an inconsistent theory, but still only containing formulas of T_1 and T_2.
But $Cn(T_1 \cup T_2)$ would now consist of all formulas, since $T_1 \cup T_2$ is inconsistent and the deductive closure contains every formula, that can be logically deducted from it (which is every formula, since it is inconsistent).
So $Cn(T_1) \cup Cn(T_2) \subseteq Cn(T_1 \cup T_2)$ holds for this case.

T_1 and T_2 do not make each other inconsistent:
In this case $Cn(T_1) \cup Cn(T_2)$ still can contain less or the same formulas as $Cn(T_1 \cup T_2)$. Joining T_1 and T_2 can only lead to more formulas being able to be deducted, than joining the deductive closures separately, since we are adding more knowledge to our theory before deduction.
So $Cn(T_1) \cup Cn(T_2) \subseteq Cn(T_1 \cup T_2)$ holds for this case too. $\square$