

COSC462: Applied Logic

exam study notes

October 24, 2004

Contents

1	Propositional Logic	3
1.1	Epistemic functioning	3
1.2	Opaque Languages	3
1.2.1	Definition of the language, L_A	3
1.2.2	Valuations, satisfaction and models	4
1.2.3	Equivalence, DNF and SDFN	5
1.2.4	Entailment	6
1.2.5	The ineffability theorem	7
1.2.6	The compactness theorem	8
1.2.7	Reasoning algorithms	9
1.3	Transparent Languages	10
1.3.1	Transparent atoms	10
1.3.2	Interpretations	10
1.3.3	Variables and pointed interpretations	10
1.3.4	Function symbols	11
1.3.5	Sorts	12
2	First-order Logic	13
2.1	Quantifiers	13
2.2	Satisfaction	13
2.3	Models	14
2.3.1	Pointed model of α	14
2.3.2	(Global) Model of α	14
2.3.3	Universal truth	14
2.4	Sentences	15
2.5	Entailment and Equivalence	15
2.5.1	Pointed	15
2.5.2	Classical/Global	15
2.5.3	Comparison	15
2.6	Situation Calculus	16

3	Defeasible Beliefs	18
3.1	Probabilistic Logic	18
3.1.1	Sample space	18
3.1.2	Conditional probabilities	19
3.1.3	Defeasible entailment	19
3.2	Nonmonotonic Logic	19
3.2.1	Ranked Interpretations	19
3.2.2	Defeasible Entailment	19
3.2.3	Belief Change	21
4	Epistemic Logic	23
4.1	Modal logic	23
4.1.1	Knowledge	23
4.1.2	Kripke Models = Possible Worlds Interpretations . . .	23
4.2	Single-agent Epistemic Logic, $\mathcal{L}_1^K(P)$	24
4.2.1	The equivalence relation	24
4.2.2	Properties of Knowledge	24
4.2.3	Kripke Frames and Correspondence Thoery	25
4.3	Multi-agent Epistemic Logic	25
4.3.1	Kripke Models and Knowledge	25
4.3.2	General Knowledge, $E_G\phi$	25
4.3.3	Common Knowledge, $C_G\phi$	25
4.3.4	Public Announcements, $[\phi]\psi$	26
5	Logic Programming	28
5.1	Clauses	28
5.2	The Martelli-Montanari unification algorithm	28
5.3	Propositional Resolution	29
5.3.1	Basic Resolution Step	29
5.3.2	Resolution Derivation or Resolution Proof	29
5.3.3	Refutation	29
5.3.4	Soundness	30
5.3.5	Completeness	30
5.3.6	General Resolution and Search Trees	30

Chapter 1

Propositional Logic

1.1 Epistemic functioning

Begin with an agent in a world whose entire set of possible states is represented by the set S .

e.g. the light-fan system, $S = \{00, 01, 10, 11\}$

stage 1: fixed information is used to limit the states to a set of candidate states, $C_f \subseteq S$. The amount of information is proportional to $\overline{C_f}$ (the complement of C_f , $S - C_f$).

e.g. There is a wire from the light switch to the fan so that if the light is on, so is the fan. Thus can eliminate state 10 to give $C_f = \{00, 01, 11\}$.

stage 2: evidence by way of observation, or communicated information, about the state of the system. Fixed information and evidence are both *definite* information, the combination giving us the candidate states C_{fe} which represents the agent's *knowledge* about the world/system.

e.g. The agent sees that the fan is on. We can thus eliminate state 00 to give $C_{fe} = \{01, 11\}$.

stage 3: default rules, heuristics or statistical information about what usually is the case. Use of this *indefinite* information leads to C_{fed} which represents the agent's *belief* about the world/system.

e.g. Usually when the fan is on, the light is on too. Thus we eliminate state 01 to give $C_{fed} = \{11\}$.

1.2 Opaque Languages

1.2.1 Definition of the language, L_A

Atoms The simplest strings in the language. Can be a finite set e.g. $A = \{p, q\}$, or an infinite set e.g. $A = \{p_0, p_1, p_2, \dots, p_\infty\}$. If A is finite,

L_A is finitely generated.

Connectives and their associated truth values:

\neg Negation. α must be false for its negation to be true.

α	$\neg\alpha$
1	0
0	1

\wedge Conjunction. Both α and β must be true for their conjunction to be true.

α	β	$\alpha \wedge \beta$
1	1	1
1	0	0
0	1	0
0	0	0

\vee Disjunction. Only one of α and β need be true for its disjunction to be true (and, in fact, they can both be true).

α	β	$\alpha \vee \beta$
1	1	1
1	0	1
0	1	1
0	0	0

\rightarrow Conditional. If the antecedent is true then the consequent must also be true for this to hold. Or in other words, either α is false (and so we don't need to look at the truth value of β) or both α and β are true.

α	β	$\alpha \rightarrow \beta$
1	1	1
1	0	0
0	1	1
0	0	1

\leftrightarrow Biconditional. If α is true then so must β be true, and if β is true then so must α be true.

α	β	$\alpha \leftrightarrow \beta$
1	1	1
1	0	0
0	1	0
0	0	1

1.2.2 Valuations, satisfaction and models

A **valuation** is an assignment of truth values (true|false, or 1|0) to all the atoms, A , in a language, L_A . The set of all possible valuations in a language, L_A , is denoted by W_A (all possible states/valuations).

e.g.1 If $A = \{p_0, p_1, \dots, p_{100}\}$, a valuation is the function $v : A \rightarrow \{0,1\}$ given by $v(x) = 1$ if x is even and $v(x) = 0$ otherwise.

e.g.2 for the p,q,r language we can simply write $W_A = \{111, 110, 101, 100, 011, 010, 001, 000\}$.

v **satisfies** α = the valuation, v , makes the sentence α true
 (by the recursive eval. of α down to the atomic level)
 = v **is a model of** α

We can also say that α is satisfiable by the valuation v . As an extension to this, a set of sentences Γ is satisfiable if there is at least one valuation that satisfies every sentence in the set.

$\mathcal{M}(\alpha)$ = The subset of W_A comprising **all models of** α .
 = Set of all states that make α true.
 = X , then we call α an **axiomatisation** of X .

$\mathcal{N}(\alpha)$ = The **nonmodels of** α .
 = The complement of the models of α , i.e. $\overline{\mathcal{M}(\alpha)}$.

tautology = a sentence α such that $\mathcal{M}(\alpha) = W_A$

contradiction = a sentence α such that $\mathcal{M}(\alpha) = \emptyset$

1.2.3 Equivalence, DNF and SDNF

α is equivalent to β (written as $\alpha \equiv \beta$) iff $\mathcal{M}(\alpha) = \mathcal{M}(\beta)$.

Equivalence is reflexive and transitive.

disjunctive normal form (DNF) A disjunction of conjunctions.

strong disjunctive normal form (SDNF) A disjunction of conjunctions where every atom in the language appears exactly once in each conjunction.

An equivalent sentence in DNF can be found for every sentence α in a language, and if that language is finite then an equivalent sentence in SDNF can be found too.

Proof of equivalence relations:

- If can see the atoms making up the sentences on either side of the relation, can simply determine the models of the each sentence as two sets of states and check their equivalence.
- If we cannot see the atoms then we must prove that if a valuation that satisfies the LHS must also satisfy the RHS, and that if a valuation satisfies the RHS it must also satisfy the LHS.

Example 1:

In the p, q language, are the following two sentences equivalent: $p \rightarrow (p \rightarrow p)$ and $(p \rightarrow p) \rightarrow p$?

- $\mathcal{M}(p \rightarrow (p \rightarrow p))$? Start with $S = \{11, 10, 01, 00\}$. Then for each state determine if the sentence is true or not: e.g. 10: p is true, and so $(p \rightarrow p)$ is also true, and so $p \rightarrow (p \rightarrow p)$ is true. End up with $\mathcal{M}(p \rightarrow (p \rightarrow p)) = \{11, 10, 01, 00\}$.
- $\mathcal{M}((p \rightarrow p) \rightarrow p)$? As above. e.g. 01: p is false so $(p \rightarrow p)$ is true, but then $(p \rightarrow p) \rightarrow p$ is now false. End up with $\mathcal{M}((p \rightarrow p) \rightarrow p) = \{11, 10\}$.
- $\mathcal{M}(p \rightarrow (p \rightarrow p)) \neq \mathcal{M}((p \rightarrow p) \rightarrow p)$ so no, these two sentences are not equivalent.

Example 1:

Is $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$?

i.e. Is it the case that $\mathcal{M}(\alpha \rightarrow \beta) = \mathcal{M}(\neg\alpha \vee \beta)$?

- Let v be a valuation that satisfies $\alpha \rightarrow \beta$ (i.e. $v \in \mathcal{M}(\alpha \rightarrow \beta)$), must it necessarily follow that this v also satisfy $\neg\alpha \vee \beta$? (i.e. must it also be the case that $v \in \mathcal{M}(\neg\alpha \vee \beta)$?). Well, v must either not satisfy α or, if it does satisfy α it must also satisfy β . In the first case v would then satisfy $\neg\alpha$ and so satisfy $\neg\alpha \vee \beta$. In the second case v satisfies β and so also satisfies $\neg\alpha \vee \beta$. Thus we can say that if v satisfies $\alpha \rightarrow \beta$ then it must also satisfy $\neg\alpha \vee \beta$.
- Let w be a valuation that satisfies $\neg\alpha \vee \beta$. Must it necessarily follow that this w must also satisfy $\alpha \rightarrow \beta$? Well, w must either satisfy $\neg\alpha$ or it must satisfy β or it must satisfy both. In the first and last case, since it satisfies $\neg\alpha$, it cannot satisfy α and so must satisfy $\alpha \rightarrow \beta$. In the second case it satisfies β but does not satisfy $\neg\alpha$, so satisfies $\alpha \rightarrow \beta$. Thus we can say that...
- So $\mathcal{M}(\alpha \rightarrow \beta) \subseteq \mathcal{M}(\neg\alpha \vee \beta)$ and $\mathcal{M}(\neg\alpha \vee \beta) \subseteq \mathcal{M}(\alpha \rightarrow \beta)$, so $\mathcal{M}(\alpha \rightarrow \beta) \equiv \mathcal{M}(\neg\alpha \vee \beta)$, so the two sentences are equivalent.

1.2.4 Entailment

α (classically) **entails** β (written as $\alpha \models \beta$) iff $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$.

Entailment is reflexive and transitive.

A set of sentences, say Γ , can also entail another sentence, say β , iff $\mathcal{M}(\Gamma) \subseteq \mathcal{M}(\beta)$, where $\mathcal{M}(\Gamma)$ is taken as the st of all valuations v such that

v satisfies every sentence in Γ .

General outline of proof for questions of the type: If $\alpha \models \beta$ then must it also be the case that $\gamma \models \delta$?:

- Assume $\alpha \models \beta$, i.e. $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$.
- Choose a valuation v that satisfies γ , i.e. $v \in \mathcal{M}(\gamma)$.
- Must it then follow, from our assumption, that v must also satisfy δ ? Can use here proof by contradiction (i.e. suppose v does not satisfy δ), or a direct proof. If this does not follow need to give a specific counter-example, e.g. make up the sentences in the p,q,r language and use a specific valuation.

Example 1:

Suppose $\alpha \models \beta$. Is it the case that $\neg\beta \models \neg\alpha$?

Assume $\alpha \models \beta$, i.e. $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$.

Let v be some valuation that satisfies $\neg\beta$, i.e. $v \in \mathcal{M}(\neg\beta)$.

Must it then be the case that v satisfies $\neg\alpha$? i.e. Does it necessarily follow that $v \in \mathcal{M}(\neg\alpha)$?

Well, suppose v does not satisfy $\neg\alpha$. Then v must satisfy α . But then following our assumption v must also satisfy β which gives us a contradiction as we chose our v to satisfy $\neg\beta$. So v cannot satisfy α so must also satisfy $\neg\alpha$.

Example 2:

Suppose $\alpha \models \beta$. Is it the case that $\alpha \wedge \gamma \models \beta$?

Assume $\alpha \models \beta$.

Let v be some valuation that satisfies $\alpha \wedge \gamma$, i.e. $v \in \mathcal{M}(\alpha \wedge \gamma)$.

Must it then be the case that v satisfies β ?

Well, since v satisfies $\alpha \wedge \gamma$ then v must satisfy both α and γ . So, from our initial assumption, v must also satisfy β .

1.2.5 The ineffability theorem

In a language, W_A represents all the possible states that can be represented by that language. If a set of states, $C \subseteq W_A$, has an axiomatisation (i.e. set of sentences Γ such that $\mathcal{M}(\Gamma) = C$) then C is **effable**. If not then C is **ineffable**.

Let $A = \{p_0, p_1, \dots\}$. There exists an ineffable set C of valuations.

i.e. for languages where the set of atoms, A , is infinite, there are some states that cannot be expressed in a sentence, or set of sentences. This is a limitation of the language.

Proof:

Pick any valuation $w \in W_A$. For example, let w be the valuation such that $w(p_i) = 1$ for all $p_i \in A$.

Now let C be all the remaining valuations, i.e. $C = \overline{\{w\}}$.

We now show that there is no set Γ of sentences of the language such that $\mathcal{M}(\Gamma) = C$, where $\mathcal{M}(\Gamma)$ is the set of all valuations that each satisfies all the sentences in Γ :

(*Proof by contradiction*)

- Pick any sentence γ which is satisfied by all valuations in C .
- There are only finitely many atoms, say p_0, \dots, p_k , in γ .
- Let $v \in C$ be a valuation which behaves like w on p_0, \dots, p_k but for some p_n , such that $n > k$, that p_n is instead false.
- Now (because in determining the satisfiability of a sentence, only the valuations of the atoms in that sentence need be taken into account) v satisfies γ iff w satisfies γ (as both valuations behave identically on the set of atoms in the sentence).
- Since we chose our γ such that it is satisfied by all valuations in C , including v , then it must be the case that w satisfies γ .
- Thus we cannot axiomatise the set of states C without also including a state, w , that is not in C .

1.2.6 The compactness theorem

A set Γ of sentences is satisfiable iff every finite subset of Γ is satisfiable.

Proof:

Forward direction of iff:

If Γ is satisfiable then there is a valuation v satisfying all the sentences in Γ , and so every finite subset of Γ is satisfied by v .

Backward direction of iff:

Suppose every finite subset of Γ is satisfiable.

1. Firstly we construct a set Δ of sentences such that:

$$\begin{aligned} & \Gamma \subseteq \Delta \\ & \text{and for all } \beta \in \mathbf{L}_A, \beta \in \Delta \text{ or } \neg\beta \in \Delta \\ & \text{and every finite subset of } \Delta \text{ must be satisfiable} \end{aligned}$$

2. Next we define a valuation $v : A \rightarrow \{0, 1\}$ by $v(p_i) = 1$ iff $p_i \in \Delta$

3. Lastly we show that for all sentences β in L_A , v satisfies β iff $\beta \in \Delta$ (using proof by induction). Thus since v satisfies all sentences in Δ it must also satisfy all sentences in Γ (since $\Gamma \subseteq \Delta$), so Γ is satisfiable.

Let Γ be a set of sentences. If $\Gamma \models \beta$ then there is some finite subset $\Gamma' \subseteq \Gamma$ such that $\Gamma' \models \beta$.

Proof:

We first show that $\Gamma \models \beta$ iff $\Gamma \cup \{\neg\beta\}$ is unsatisfiable:

- Suppose $\Gamma \models \beta$. So any valuation that satisfies all the sentences in Γ must also satisfy β . So there is no valuation that can satisfy all the sentences in Γ and satisfy $\neg\beta$ (by failing to satisfy β). i.e. $\Gamma \cup \{\neg\beta\}$ is unsatisfiable.
- Suppose $\Gamma \cup \{\neg\beta\}$ is unsatisfiable. So every valuation that satisfies Γ must fail to satisfy $\neg\beta$, thereby satisfying β . i.e. $\Gamma \models \beta$

Using this and the compactness theorem we finalize the proof:

- Suppose $\Gamma \models \beta$. Then, from above, $\Gamma \cup \{\neg\beta\}$ is unsatisfiable. Thus, using the compactness theorem, there must be a finite subset, say $\Gamma' \cup \{\neg\beta\}$ (where $\Gamma' \subseteq \Gamma$) that is also unsatisfiable. So, if this Γ' is satisfied by some valuation v then v must fail to satisfy $\neg\beta$ and so must satisfy β . i.e. $\Gamma' \models \beta$.

1.2.7 Reasoning algorithms

Database: Set of sentences Γ

Query: β

“Is it the case that $\Gamma \models \beta$ ”?

An automated reasoner uses *inferences* to deduce a sentence from a set of sentences. *Modus ponens* (MP) is a classic type of inference: given $\alpha \rightarrow \beta$ and α , then we can deduce β . Furthermore we can do this recursively back from β :

$\Gamma \vdash \beta$ = there exists a MP-deduction of β from Γ
 = Γ is a sequence of sentences $\langle \alpha_0, \dots, \alpha_n \rangle$ such that $\alpha_n = \beta$ and for all $k \leq n$ either $\alpha_k \in \Gamma$ or there is a sentence which implies α_k (i.e. there exists $i, j < k$ such that $\alpha_j = (\alpha_i \rightarrow \alpha_k)$).

The reasoner is sound (if $\Gamma \vdash \beta$ then $\Gamma \models \beta$) but not complete (if $\Gamma \models \beta$ it is not always the case that $\Gamma \vdash \beta$, e.g. $\Gamma = \{p, q\}$, we know $\Gamma \models p \wedge q$ but this is not able to be deduced by the reasoner).

1.3 Transparent Languages

1.3.1 Transparent atoms

Cons constant symbols nouns e.g. light, fan, c, b

Pred predicate symbols verbs e.g. IsOn/1, P/2

e.g. the light-fan-heater system where each component can be on|off or functioning|defective:

Cons = {*Light, Fan, Heater*}
Pred = {(*IsOn*, 1), (*IsFunctioning*, 1)}
Atoms, A = {*IsOn(Light), IsOn(Fan), IsOn(Heater), IsFunctioning(Light), IsFunctioning(Fan), IsFunctioning(Heater)*}

1.3.2 Interpretations

An interpretation $\mathcal{D} = (D, den)$, where D is the domain and den is the denotation function which assigns to each $c \in Cons$ a member $den(c) \in D$ and to each $(P, n) \in Pred$ a subset $den(P, n)$ of D^n .

A **term** interpretation is one in which $den(c) = c$, i.e. the constant symbols themselves make up the domain and so map to themselves.

e.g. The horses *Andy* and *Bandy* who either do or donot require feeding or grooming. Describe a term interpretation where both horses need grooming but only *Andy* needs feeding.

Cons = {*Andy, Bandy*}
Pred = {(*NeedsFeeding*, 1), (*NeedsGrooming*, 1)}

Term interpretation $\mathcal{D} = (D, den)$ where $D = Cons$, and $den(c) = c$ for all $c \in Cons$. e.g. $den(Andy) = Andy$

$den(NeedsFeeding, 1) = Andy$
 $den(NeedsGrooming, 1) = Andy, Bandy$

Suppose we know that *Bandy* only ever needs grooming when *Andy* needs grooming. We can thus eliminate any term interpretations where $Andy \notin den(NeedsGrooming, 1)$ but $Bandy \in den(NeedsGrooming, 1)$.

1.3.3 Variables and pointed interpretations

Variables are a way of representing pronouns, e.g. it, he, she.

Var = { x_1, x_2, \dots }
a term = a constant or variable
Term = $Cons \cup Var$

Atoms, A = all strings $P(t_1, t_2, \dots, t_n)$ such that $(P, n) \in Pred$ and $t_1, t_2, \dots, t_n \in Term$.

e.g. Let $Cons = \{a, b\}$ and $Pred = \{(P, 1)\}$. Then $Term = \{a, b, x_1, x_2, \dots\}$

and the set of atomic formulae is $A = \{P(a), P(b), P(x_1), P(x_2), \dots\}$.

A **pointed interpretation** is an interpretation \mathcal{D} together with a **variable assignment** (a function $s : Var \rightarrow D$).

e.g. Using the above example ($Cons = \{a, b\}$ and $Pred = \{(P, 1)\}$), a possible interpretation is $\mathcal{D} = (D, den)$ where $D = \{42, 69, 112\}$, $den(a) = 42$, $den(b) = 69$ and $den(P, 1) = \{42, 112\}$. Now we add a variable assignment, $s : Var \rightarrow D$ where $s(x_i) = 112$ if i is even and 69 otherwise.

The valuation determined by the pointed interpretation (\mathcal{D}, s) is the function $v : A \rightarrow \{0, 1\}$ such that for every $P(t_1, t_2, \dots, t_n) \in A$, $v(P(t_1, t_2, \dots, t_n)) = 1$ iff $(d_1, d_2, \dots, d_n) \in den(P, n)$ where $d_i = den(t_i)$ if $t_i \in Cons$ or $d_i = s(t_i)$ if $t_i \in Var$.

i.e. den finds the denotations of constants and s the denotation of variables.

e.g. Continuing the above example, we calculate the following evaluation:

Since $den(a) = 42 \in den(P, 1)$, we can say $v(P(a)) = 1$

Since $den(b) = 69 \notin den(P, 1)$, we can say $v(P(b)) = 0$

And, for even i , $s(x_i) = 112 \in den(P, 1)$ so $v(P(x_i)) = 1$ (for even i)

And, for odd i , $s(x_i) = 69 \notin den(P, 1)$ so $v(P(x_i)) = 0$ (for odd i)

So we can say that our variable assignment, s , satisfies $P(x_4)$ but fails to satisfy $P(x_4) \wedge P(x_5)$.

A **pointed model** of a sentence α is a pointed interpretation (\mathcal{D}, s) where the variable assignment $s : Var \rightarrow D$ satisfies α in \mathcal{D} . $\mathcal{PM}(\alpha)$ = the set of all pointed models of α .

e.g. Continuing the above example: (\mathcal{D}, s) is a pointed model of the sentence $P(a) \wedge P(x_4)$.

A **model** of a sentence α is an interpretation \mathcal{D} where every variable assignment $s : Var \rightarrow D$ satisfies α in \mathcal{D} . $\mathcal{M}(\alpha)$ = the set of all models of α .

e.g. Continuing the above example: \mathcal{D} is a model of the sentence $P(b)$.

$\alpha \models \beta$ iff $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$ and $\alpha \models^p \beta$ iff $\mathcal{PM}(\alpha) \subseteq \mathcal{PM}(\beta)$

1.3.4 Function symbols

Used to represent definite descriptions, e.g. the capital of Sri Lanka, or the mother of Hau.

Fun = set of ordered pairs of type (f, n)

$Term$ consists of all strings t such that one of the following is the case:

- $t \in Cons$
- $t \in Var$
- $t = f(t_1, t_2, \dots, t_n)$ where $(f, n) \in Fun$ and t_1, t_2, \dots, t_n are previously generated terms.

To determine the truth of a function, must first use den to denotate its atomic terms and s to denotate its variable terms. Then we use den to denotate the function itself.

e.g. A function $(SwitchOf, 1)$ with the instance $SwitchOf(Light)$ can be denotated as the light switch.

1.3.5 Sorts

Used to determine what terms a function can act on and what terms it can produce. All objects in the domain must belong to one, and only one, sort. For example, a function f of arity 2 may act on two different sorts in the domain, returning a third:

$$den(f, 2) : D_{sort1}, D_{sort2} \rightarrow D_{sort3}$$

Chapter 2

First-order Logic

2.1 Quantifiers

For languages with variables (the set Var):

Connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

Quantifiers: $\forall x_i$ (for all), $\exists x_i$ (for some, there exists)

e.g. notice the difference between “there exists a plate for all the guests” and “for all of the guests there is a plate”: in the first sentence we can think that there is only one big plate for everyone, and in the second that there is a plate for each guest.

2.2 Satisfaction

s satisfies $\exists x_i(\alpha)$ in (D, den) iff
 s' satisfies α in (D, den) for some $s' : Var \rightarrow D$ that differs from s at most on x_i

s satisfies $\forall x_i(\alpha)$ in (D, den) iff
 s' satisfies α in (D, den) for every $s' : Var \rightarrow D$ that differs from s at most on x_i

To determine if an atom $P(t_1)$ is satisfied we first need either an a denotation function den or a variable assignment s or both. This converts the term t_1 into an element in the domain (i.e. $d_1 \in D$). Then the problem becomes: is $d \in den(P, 1)$? If yes, then $P(t_1)$ is satisfied.

We can use the exists operation to expand possibilities and the for all operator to limit possibilities:

e.g.1 Using the equality predicate $(=, 2)$, give a sentence that has at least two items in its domain: $\exists x_1 \exists x_2 (\neg(x_1 = x_2))$; give a sentence which has exactly two items in its domain: $\exists x_1 \exists x_2 (\neg(x_1 = x_2) \wedge \forall x_3 ((x_3 = x_1) \vee (x_3 = x_2)))$; give a sentence that has at most two items in its domain: $\exists x_1 \exists x_2 \forall x_3 ((x_3 =$

$x_1) \vee (x_3 = x_2)$; we test this using a domain containing two elements, e.g. $D = \{1, 2\}$, $s(x_i) = 1$ for odd i 's and 2 for even i 's, and of course $\text{den}(=, 2) = \{(1, 1), (2, 2)\}$. We just need one of the following boxes to contain all True values:

	x_1	x_2	x_3	
s	1	2	1	
s'	1, or	1, or	1, and	T
s'			2	F
s'		2	1, and	T
s'			2	T
s'	2	1, or	1, and	T
s'			2	T
s'		2	1, and	F
s'			2	T

We should then test using a domain containing just one element, e.g. $D = \{1\}$ but this is trivial as there is only one possible variable assignment and that results in a True outcome. Lastly we should check that this sentence will not be satisfied with a domain of three elements, e.g. $D = \{1, 2, 3\}$. Here we can see that no matter what we assign to x_1 and x_2 we can always find a value for x_3 that is different from both of these.

2.3 Models

2.3.1 Pointed model of α

A **pointed model** of α is an interpretation ($\mathcal{D} = (D, \text{den})$) and a variable assignment ($s : \text{Var} \rightarrow D$) such that s satisfies α in \mathcal{D} .

2.3.2 (Global) Model of α

A **global model** of α is an interpretation ($\mathcal{D} = (D, \text{den})$) such that every variable assignment $s : \text{Var} \rightarrow D$ satisfies α in \mathcal{D} .

2.3.3 Universal truth

A wff α is **universally true** if it is satisfied by every interpretation by all assignments.

e.g. Is the wff $\exists x_2(P(x_2)) \rightarrow \neg \forall x_2 \neg P(x_2)$ satisfied by every assignment in every interpretation?

- Assume we have an interpretation \mathcal{D} and a variable assignment s that satisfies $\exists x_2(P(x_2))$. Does it necessarily follow that (\mathcal{D}, s) must also satisfy $\neg\forall x_2\neg P(x_2)$?
- Well, since s satisfies $\exists x_2(P(x_2))$ then we know that there must exist some element from the domain, say d_1 , where $s(x_2) = d_1$ and $d_1 \in \text{den}(P, 1)$, i.e. $\text{den}(P, 1)$ is non-empty.
- Suppose that this s does not satisfy $\neg\forall x_2\neg P(x_2)$. It follows that it must then satisfy $\forall x_2\neg P(x_2)$. So, all variable assignments must fail to satisfy $P(x_2)$. But we know that $\text{den}(P, 1)$ is non-empty, therefore there must exist at least one assignment, s' , that maps x_2 to d_1 and in doing so satisfies $P(x_2)$. Thus we have a contradiction and conclude that this s must satisfy $\neg\forall x_2\neg P(x_2)$.

2.4 Sentences

A sentence is a **closed well-formed formula**, i.e. every variable in the wff is in the scope of a quantifier which binds it

2.5 Entailment and Equivalence

2.5.1 Pointed

$$\begin{aligned} \alpha \models^p \beta &\text{ iff } \mathcal{PM}(\alpha) \subseteq \mathcal{PM}(\beta) \\ \text{i.e. every model } \mathcal{D} &\text{ of } \alpha \text{ is also a model of } \beta \\ \alpha \equiv^p \beta &\text{ iff } \mathcal{PM}(\alpha) = \mathcal{PM}(\beta) \end{aligned}$$

2.5.2 Classical/Global

$$\begin{aligned} \alpha \models \beta &\text{ iff } \mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta) \\ \text{i.e. every pointed model } (\mathcal{D}, s) &\text{ of } \alpha \text{ is also a pointed model of } \beta \\ \alpha \equiv \beta &\text{ iff } \mathcal{M}(\alpha) = \mathcal{M}(\beta) \end{aligned}$$

2.5.3 Comparison

Pointed entailment (\models^p) is a stricter form of entailment than classical entailment (\models).

e.g. $P(x_1) \models \forall x_1 P(x_1)$ but $P(x_1) \not\models^p \forall x_1 P(x_1)$

If an interpretation models $P(x_1)$ then all possible variable assignments must satisfy $P(x_1)$. Thus we can see that these models will also satisfy $\forall x_1 P(x_1)$ as this simply requires every possible variable assignment of x_1 satisfy $P(x_1)$.

However, a pointed interpretation may satisfy $P(x_1)$ but fail to satisfy $\forall x_1 P(x_1)$. Let $\mathcal{D} = (D, \text{den})$ be the interpretation such that $D = \{1, 2, 3, \dots\}$ and $\text{den}(P, 1) = \{1\}$. Then we have a variable assignment, $s : \text{Var} \rightarrow D$ such that $s(x_i) = 1$ when i is odd, and $s(x_i) = 2$ when i is even. Since $s(x_1) = 1 \in \text{den}(P, 1)$ then this s satisfies $P(x_1)$. However, for s to satisfy $\forall x_1 P(x_1)$ there must not exist an assignment s' that behaves exactly like s on all variables except possibly x_1 . Let this $s' : \text{Var} \rightarrow D$ be such that $s(x_i) = 2$ for all i . Then $s'(x_1) = 2 \notin \text{den}(P, 1)$ and so s does not satisfy $\forall x_1 P(x_1)$.

2.6 Situation Calculus

Introduces three sorts: Fact, Action, and Situation; Requires the predicate (Holds, 2) fo sort $\langle \text{Fact}, \text{Situation} \rangle$. This gives us our link to determining truth. Requires the function Result of sort $\langle \text{Action}, \text{Situation}, \text{Situation} \rangle$: i.e. Result takes as arguments an Action in a particular Situation and returns the resulting Situation.

e.g. the Light-Fan-Switch system where there is one light, one fan and a separate switch for each of them:

$$\begin{aligned}
 \text{Sorts} &= \{ \text{Situation, Fact, Action,} \\
 &\quad \text{Component, Switch} \} \\
 \text{Cons} &= \{ S_0, S_L, S_F, S_{LF} \text{ of sort Situation,} \\
 &\quad F_L, F_F \text{ of sort Fact,} \\
 &\quad A_{PL}, A_{PF} \text{ of sort Action,} \\
 &\quad C_L, C_F \text{ of sort Component,} \\
 &\quad Sw_L, Sw_F \text{ of sort Switch} \} \\
 \text{Var} &= \{ s_1, s_2, \dots \text{ of sort Situation,} \\
 &\quad f_1, f_2, \dots \text{ of sort Fact,} \\
 &\quad a_1, a_2, \dots \text{ of sort Action,} \\
 &\quad c_1, c_2, \dots \text{ of sort Component,} \\
 &\quad sw_1, sw_2, \dots \text{ of sort Switch} \} \\
 \text{Fun} &= \{ \text{Result of sort } \langle \text{Action, Situation, Situation} \rangle, \\
 &\quad \text{SwitchOf of sort } \langle \text{Component, Switch} \rangle, \\
 &\quad \text{IsOn of sort } \langle \text{Component, Fact} \rangle, \\
 &\quad \text{IsFunctioning of sort } \langle \text{Component, Fact} \rangle, \\
 &\quad \text{Press of sort } \langle \text{Switch, Action} \rangle \} \\
 \text{Pred} &= \{ (\text{Holds, 2}) \text{ of sort } \langle \text{Fact, Situation} \rangle \}
 \end{aligned}$$

Of course, because we defined the function SwitchOf then we do not strictly

need the Switch constants Sw_L and Sw_F ; and likewise, because we defined the function *Press* we do not need the Action constants A_{PL} and A_{PF} ; and finally, because we defined the function *IsOn* we may do away with the Facts F_L and F_F . I include them here just for completeness and flexibility.

Now, to say that if a component is on and we press its switch it turns it off we would write the sentence:

$$\forall c_1 s_1 (\text{Holds}(\text{IsOn}(c_1), s_1) \rightarrow \text{Holds}(\neg \text{IsOn}(c_1), \text{Result}(\text{Press}(\text{SwitchOf}(c_1)), s_1)))$$

Sometimes we may want to say that if an initial setup results in an action that is impossible then trying to perform that action should have no effect on the system. To do this we may add another predicate symbol: the equality operator, i.e. $(=, 2)$ of sort $\langle \text{Situation}, \text{Situation} \rangle$. Then we can say that performing the action in the situation results in the same situation, i.e. nothing is changed.

Chapter 3

Defeasible Beliefs

from \models classical entailment
to \vdash defeasible entailment

3.1 Probabilistic Logic

3.1.1 Sample space

Probability space $\langle S, \mathcal{B}, Pr \rangle$:

- S = a nonempty set, the **sample space**
 - = $\{s_1, \dots, s_n\}$ such that s_i are **outcomes**
 - = *e.g.* $S = \{H, T\}$
- \mathcal{B} = a field of subsets of S
 - = $\{b_1, \dots, b_n\}$ such that s_i are **events**
 - (obtained from **elementary events** using \cup, \cap , and complement, and including S and \emptyset)
 - = *e.g.* $\mathcal{B} = \{\emptyset, \{H\}, \{T\}, \{H, T\}\}$
- Pr = a probability measure on \mathcal{B}
 - : $\mathcal{B} \rightarrow [0, 1]$
 - = *e.g.* $Pr(\emptyset) = 0, Pr(\{H\}) = \frac{1}{2}, Pr(\{T\}) = \frac{1}{2}, Pr(\{H, T\}) = 1$

To work out the initial probability of a sentence α , determine $Pr(\mathcal{M}(\alpha))$. $\mathcal{M}(\alpha)$ gives a set of states/outcomes that will be in the set \mathcal{B} and since $Pr : \mathcal{B} \rightarrow [0, 1]$ we will have Pr .

e.g. the light-fan system. $S = \{11, 10, 01, 00\}$, so $\mathcal{B} = \{\emptyset, \{11\}, \{10\}, \{01\}, \{00\}, \{11, 10\}, \dots, \{11, 10, 01\}, \dots, \{11, 10, 01, 00\}\}$. If we take the probability of each of the possible four states, the elementary events, to be $\frac{1}{4}$, then:

$$\begin{aligned} Pr(p \vee \neg q) &= Pr(\mathcal{M}(p \vee \neg q)) \\ &= Pr(\{11, 10, 00\}) \\ &= \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = \frac{3}{4}. \end{aligned}$$

3.1.2 Conditional probabilities

Given some event E as expressed by the sentence β , determine the probability of α :

$$\begin{aligned} Pr(\alpha | \beta) &= \frac{Pr(\alpha \wedge \beta)}{Pr(\beta)} \\ &= \frac{Pr(\mathcal{M}(\alpha) \cap \mathcal{M}(\beta))}{\mathcal{M}(\beta)} \end{aligned}$$

3.1.3 Defeasible entailment

Threshold = our confidence in our conclusion
 $\in [0,1]$

$$\alpha \sim \beta \text{ iff } Pr(\beta | \alpha) \geq \textit{threshold}$$

3.2 Nonmonotonic Logic

3.2.1 Ranked Interpretations

\preceq is a **preorder** on a set S iff:

- \preceq is reflexive on S , and
- \preceq is transitive on S

\preceq is a **total preorder** on a set S iff:

- \preceq is a preorder on S , and
- \preceq ranks **all** possible pairs of elements in S

A **(finite) ranked interpretation**, \mathcal{I} is the tuple $\langle S, \preceq, V \rangle$ such that:

- S is a (finite) non-empty set of states/possible worlds
- \preceq is a total prorder on S
- $V : S \rightarrow W_A$ is a **labelling function** which maps the set of states to a set of valuations of the language; sometimes it is the identity function

Hass diagrams verses filing cabinet diagrams. Normality increases the lower we go and $s_1 \preceq s_2$ then s_1 is more likely than s_2 .

3.2.2 Defeasible Entailment

$$\alpha \sim \beta \text{ iff } \textit{Min}(\alpha) \subseteq \mathcal{M}(\beta)$$

Cautious monotonicity property

if $\alpha \sim \beta$ and $\alpha \sim \gamma$ then $\alpha \wedge \gamma \sim \beta$

Proof:

Assume $\alpha \sim \beta$ and $\alpha \sim \gamma$, i.e. $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta)$ and $\text{Min}(\alpha) \subseteq \mathcal{M}(\gamma)$. Does it necessarily follow that $\alpha \wedge \gamma \sim \beta$?

Pick any state s such that $s \in \text{Min}(\alpha \wedge \gamma)$. Is $s \in \mathcal{M}(\beta)$?

Suppose $s \notin \mathcal{M}(\beta)$.

Since $s \in \text{Min}(\alpha \wedge \gamma)$ we know that $s \in \mathcal{M}(\alpha)$. If this s was also a minimal model of α then we could conclude that it would be a model of β too, contradicting our statement that s was not a model of β and we would be done. So, if $s \notin \text{Min}(\alpha)$ then there would have to exist some state s' such that $s' \preceq s$ and $s' \in \text{Min}(\alpha)$. It would then follow from $\text{Min}(\alpha) \subseteq \mathcal{M}(\gamma)$ that this $s' \in \mathcal{M}(\gamma)$ too. Thus $s \in \mathcal{M}(\alpha \wedge \gamma)$. But this then contradicts our choice of s such that $s \in \text{Min}(\alpha \wedge \gamma)$ since $s' \preceq s$. So $s \in \text{Min}(\alpha)$ and it then follows that $s \in \mathcal{M}(\beta)$.

Supraclassical property

if $\alpha \models \beta$ then $\alpha \sim \beta$

Proof:

$\alpha \models \beta$ means that $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$. And since, obviously, $\text{Min}(\alpha) \subseteq \mathcal{M}(\alpha)$ then $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta)$. Thus $\alpha \sim \beta$.

(And) property

If $\alpha \sim \beta$ and $\alpha \sim \gamma$ then $\alpha \sim \beta \wedge \gamma$.

Proof:

If $\alpha \sim \beta$ and $\alpha \sim \gamma$, then is it necessarily the case that $\alpha \sim \beta \wedge \gamma$?

In other words, if $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta)$ and $\text{Min}(\alpha) \subseteq \mathcal{M}(\gamma)$, does it follow that $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta \wedge \gamma)$?

Well, let us choose an s such that $s \in \text{Min}(\alpha)$. Thus $s \in \mathcal{M}(\beta)$ and $s \in \mathcal{M}(\gamma)$, so $s \in \mathcal{M}(\beta \wedge \gamma)$.

Monotonicity does not hold

It is not the case that if $\alpha \sim \beta$ then $\alpha \wedge \gamma \sim \beta$.

Proof:

i.e. if $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta)$ does it necessarily follow that $\text{Min}(\alpha \wedge \gamma) \subseteq \mathcal{M}(\beta)$?

Well, no. Take for example the p, q system where $01 \preceq 00 \preceq 10 \preceq 11$ resulting in the filing cabinet diagram below:

11
10
00
01

Now let $\alpha = \neg q$, $\beta = \neg p$ and $\gamma = p$. Thus $\mathcal{M}(\beta) = \{01, 00\}$. We can see that $\text{Min}(\alpha) = \{00\} \subseteq \mathcal{M}(\beta)$ but that $\text{Min}(\alpha \wedge \gamma) = \{10\} \not\subseteq \mathcal{M}(\beta)$.

Transitivity does not hold

It is not the case that if $\alpha \sim \beta$ and $\beta \sim \gamma$ that $\alpha \sim \gamma$.

Proof:

i.e. if $\text{Min}(\alpha) \subseteq \mathcal{M}(\beta)$ and $\text{Min}(\beta) \subseteq \mathcal{M}(\gamma)$, does it necessarily follow that $\text{Min}(\alpha) \subseteq \mathcal{M}(\gamma)$?

Well, no. Take for example the p, q system and the same preorder as described above. Now let $\alpha = p$, $\beta = \neg q$ and $\gamma = \neg p$. $\mathcal{M}(\alpha) = \{11, 10\}$, $\mathcal{M}(\beta) = \{10, 00\}$ and $\mathcal{M}(\gamma) = \{01, 00\}$. And we can see that $\text{Min}(\alpha) = \{10\} \subseteq \mathcal{M}(\beta)$ and $\text{Min}(\beta) = \{00\} \subseteq \mathcal{M}(\gamma)$, but that $\text{Min}(\alpha) = \{10\} \not\subseteq \mathcal{M}(\gamma)$.

3.2.3 Belief Change

Recall definite information leads to C_{fe} and indefinite information leads to C_{fed} . We can talk about an agent's knowledge by giving the set of sentences that are consequences of the axiomatisation of the minimum models of a preorder, i.e. those states in the bottom drawer of the filing cabinet.

Lexicographic refinement of \preceq (definite information) by \sqsubseteq (default rules)

1. apply first ordering to give coarse grouping
2. refine by going into each drawer and apply the second ordering

e.g. A total preorder on the light-fan, $S = \{11, 10, 01, 00\}$, where the agent can see that the light is on (its definite information), followed by the default rule that it is usually the case that the light and fan are usually on (or off) together, rather than just one of the components being on, i.e. the lexicographic refinement of $11, 10 \preceq 01, 00$ by $11, 00 \sqsubseteq 10, 01$:

01	00	becomes	01
11	10		00
			10
			11

The Belief Set, K

The belief set is closed under entailment \models ,

$$\begin{aligned} \text{i.e. } K &= \{\beta \mid K \models \beta\} \\ &= \{\beta \mid \text{all the sentences that are satisfied by the valuation/states in the bottom drawer}\} \\ &= C_n(K) \end{aligned}$$

where $C_n(K)$ are all the consequences (supersets) of K .

e.g. Let an agent's belief set K be the sentences satisfied by states in the bottom level of the lexicographic refinement above.

$$K_{fed} = \{\beta \mid \{11\} \subseteq \mathcal{M}(\beta)\} = C_n(p \wedge q) = \{p \wedge q, p, q, p \vee q, p \vee \neg q, \neg p \vee q, p \leftrightarrow q, p \vee \neg p, q \vee \neg q\}.$$

Expansion, contraction and revision:

$$\begin{aligned} K + \phi &\text{ is a set of all sentences satisfied by } \mathcal{M}(K) \cup \phi \\ K - \phi &\text{ is a set of all sentences satisfied by } \mathcal{M}(K) \cup \text{Min}(\neg\phi) \\ K * \phi &\text{ is a set of all sentences satisfied by } \text{Min}(\phi) \end{aligned}$$

Expansion only works when the information we are adding is purely additional, i.e. it does not contradict in any way our previous held beliefs. e.g. if we know p and then are told q we can use expansion.

e.g. Let $K_{fed} = \{\beta \mid \{11\} \subseteq \mathcal{M}(\beta)\}$ from above:

$$\begin{aligned} K - p &= (\text{all sentences satisfied by } \mathcal{M}(K) \cup \text{Min}(\neg p) = \{11, 00\}) \\ &= C_n(p \leftrightarrow q) \\ K * (\neg p \vee \neg q) &= (\text{all sentences satisfied by } \text{Min}(\neg p \vee \neg q) = \{10\}) \\ &= C_n(p \wedge \neg q). \end{aligned}$$

Chapter 4

Epistemic Logic

4.1 Modal logic

4.1.1 Knowledge

$K\alpha$	=	$\Box\alpha$	=	agent knows α
$M\alpha$	=	$\Diamond\alpha$	=	$\neg K\neg\alpha$
			=	agent can think it is possible that α

An agent at least knows its own state.

e.g. There are three cards (0,1 and 2) and one player, Anne. We can say that once the cards are dealt:

$$K0_A \vee K1_A \vee K2_A$$

4.1.2 Kripke Models = Possible Worlds Interpretations

A Kripke Model $M = \langle S, R, V \rangle$:

- S = all factual possible states, domain of worlds
- R = an accessibility relation $R \subseteq S \times S$
 - = is reflexive and transitive
- V = (or Π) the valuation or labelling function
 - : $S \times A \rightarrow \{true, false\}$
- or : $A \rightarrow S$, *e.g.* $V(p) = \{11, 10\}$

Connectives = $\{\neg, \wedge, K\} \equiv \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, K, M\}$:

$M, s \models p$ iff $V_s(p) = true$ (or $s \in V(p)$)

$M, s \models \neg\alpha$ iff M, s fails to satisfy α

$M, s \models \alpha \wedge \beta$ iff $M, s \models \alpha$ and $M, s \models \beta$

$M, s \models K\alpha$ iff $\forall s' : R(s, s'), K, s' \models \alpha$

(note: can read the \models above as *satisfies*)

4.2 Single-agent Epistemic Logic, $\mathcal{L}_1^K(P)$

4.2.1 The equivalence relation

Let R be the **equivalence relation**, \sim . Now $(M, s) : s \in S$ is an epistemic, or information, state.

$$M, s \models K\alpha \text{ iff } \forall s' : s \sim s, M, s' \models \alpha$$

So, s is the current state of the system. Thus we say that an agent knows a sentence α to be true only if α is true in all the equivalent states (in the agents mind) of the system.

Three cards (0,1 and 2). One to be dealt to Anne, one placed on the table and one to remain in the stack holder. Since we will have 6 possible states we will call our Kripke Model, or Possible Worlds Interpretation, of this Hexa1. Anne will know her own card but will not know the card on the table or in the stack.

$$\text{Hexa1} = \langle S, \sim, \Pi \rangle$$

$$S = \{ 012, 021, 102, 120, 201, 210 \}$$

$$\sim = \{ (012, 012), (012, 021), (021, 012), (021, 021), \\ (102, 102), (102, 120), (120, 102), (120, 120), \\ (201, 201), (201, 210), (210, 201), (210, 210) \}$$

$$A = \{ 0_A, 0_t, 0_s, 1_A, 1_t, 1_s, 2_A, 2_t, 2_s \}$$

We can represent the valuation function in one of two ways. Either for each state and for each atom give a valuation of true or false: i.e. $\Pi_{012}(0_A) = \text{true}$, $\Pi_{012}(0_t) = \Pi_{012}(0_s) = \text{false}$ etc. Or, for each atom give the set of state which satisfy it: i.e. $\Pi(0_A) = \{012, 021\}$, $\Pi(0_t) = \{102, 201\}$ etc.

Now, we can show that Anne knows she has card 0. Hexa1, $012 \models K0_A$ iff $\forall s' : 012 \sim s', \text{Hexa1}, s' \models 0_A$. Well, lets check those s' states, i.e. 012 and 021.

Does $012 \models 0_A$? $0_A \in \Pi_{012}$ so yes.

Does $021 \models 0_A$? $0_A \in \Pi_{021}$ so yes.

4.2.2 Properties of Knowledge

What you know, is true (truth axiom):

$$K\phi \rightarrow \phi$$

You are aware of your knowledge (positive introspection):

$$K\phi \rightarrow KK\phi$$

You are aware of your ignorance (negative introspection):

$$\neg K\phi \rightarrow K\neg K\phi$$

4.2.3 Kripke Frames and Correspondence Theory

A Kripke Frame is a Kripke Model less the valuation function. i.e. it is just $\langle S, R \rangle$.

$K\phi \rightarrow \phi$ is valid on a frame iff the frame is **reflexive**.

$K\phi \rightarrow KK\phi$ is valid on a frame iff the frame is **transitive**.

$\neg K\phi \rightarrow K\neg K\phi$ is valid on a frame iff the frame is **euclidean**.

4.3 Multi-agent Epistemic Logic

4.3.1 Kripke Models and Knowledge

$$\begin{aligned} M, s \models K_n\phi & \text{ iff } \forall s' : s \sim_n s', s' \models \phi \\ M, s \models M_n\phi & \text{ iff } \exists s' : s \sim_n s' \text{ and } s' \models \phi \end{aligned}$$

4.3.2 General Knowledge, $E_G\phi$

$$\begin{aligned} E_G\phi & = K_1\phi \wedge K_2\phi \wedge \dots \wedge K_M\phi \\ & = \bigwedge_{m \in G} K_m\phi \end{aligned}$$

Everyone in the group G knows ϕ .

4.3.3 Common Knowledge, $C_G\phi$

$$\begin{aligned} C_G\phi & = \phi \wedge E_G\phi \wedge E_G E_G\phi \wedge \dots \\ & = \phi \wedge K_1\phi \wedge K_2\phi \wedge K_1 K_1\phi \wedge K_1 K_2\phi \wedge \dots \end{aligned}$$

Everyone in the group G knows ϕ , and everyone knows everyone knows ϕ and everyone knows everyone knows everyone knows ϕ and so on.

Transitive Closure

$$\sim_G = \left(\bigcup_{n \in G} \sim_n \right)^* = R^*$$

$$M, s \models C_G \phi \text{ iff } \forall s' : s \sim_G s', M, s' \models \phi$$

e.g. The four state system, $S = \{w, x, y, z\}$, where $w \sim_3 x$, $x \sim_2 v$ and $x \sim_1 y$, and $V(p) = \{v, x\}$ and $V(q) = \{w, v\}$. Now, we can show that $M, w \models C_{2,3}(p \rightarrow K_2 p)$.

$M, w \models C_{23}(p \rightarrow K_2 p)$ iff $\forall s : w \sim_{23} s, s \models (p \rightarrow K_2 p)$.

Well, $w \sim_{23} w$, $w \sim_{23} x$ and $w \sim_{23} v$, so let us now check each state.

$s = w$ Here p is not satisfied ($w \notin V(p)$) so $p \rightarrow K_2 p$ is satisfied.

$s = x$ $x \in V(p)$ so p is satisfied. Now $x \models K_2 p$ iff $\forall s' : x \sim_2 s', s' \models p$.

Well, $x \sim_2 x$ and $x \sim_2 v$.

$s' = x$ Since $x \in V(p)$ we know $x \models p$.

$s' = v$ Since $v \in V(p)$ we know $v \models p$.

So, x also satisfies $K_2 p$.

$s = v$ $v \in V(p)$ so p is satisfied. $v \sim_2 v$ and $v \sim_2 x$. Both v and x satisfy p so v satisfies $K_2 p$.

Since all states (2,3) accessible from state w satisfy the sentence $p \rightarrow K_2 p$ then $M, w \models C_{2,3}(p \rightarrow K_2 p)$.

4.3.4 Public Announcements, $[\phi]\psi$

Pruning the state space. If the public announcement, ϕ is true then remove all states from the state space where ϕ cannot be satisfied. Now, with the new Kripke model $M' = M \upharpoonright \phi$, does ψ still hold?

$$M, s \models [\phi]\psi \text{ iff } M, s \models \phi \text{ implies } M \upharpoonright \phi, s \models \psi$$

$$M, s \models \langle \phi \rangle \psi \text{ iff } M, s \models \phi \text{ and } M \upharpoonright \phi, s \models \psi$$

e.g.1 Hans announces to the class: “you don’t know that I have a kowhai in my garden”. Let this be represented by $p \wedge \neg K_{GP}$. But after this announcement the sentence is no longer true.

i.e. $\neg(p \wedge \neg K_{GP}) = \neg p \vee K_{GP}$

Before the announcement there are two possible states: state s where p is satisfied, and state t where p is not satisfied, so $\neg p$ is. So, before the announcement:

$$M, s \models p \wedge \neg K_{GP}$$

$$M, t \not\models p \wedge \neg K_{GP}$$

After the announcement we now know Hans has a kowhai tree in his garden because we trust he is telling us the truth:

$$M', s \models K_{GP}$$

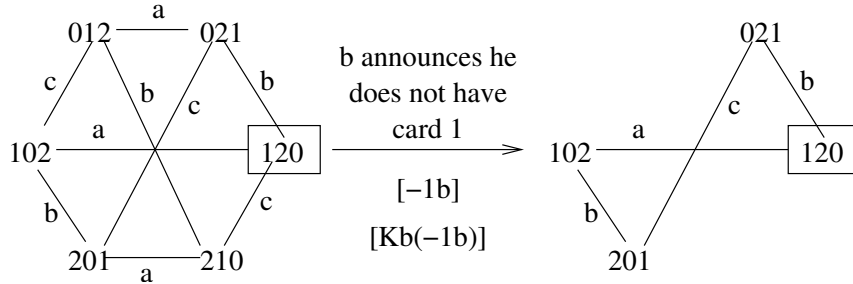
$$M', s \not\models p \wedge \neg K_{GP}$$

After the announcement we can check that our assumption that $p \wedge \neg K_{GP}$ is no longer true:

$$M, s \models [p \wedge \neg K_{GP}] \neg(p \wedge \neg K_{GP})$$

$$\text{iff } M, s \models p \wedge \neg K_{GP} \text{ implies } M \mid p \wedge \neg K_{GP}, s \models \neg(p \wedge \neg K_{GP})$$

e.g.2 Hexa where the state is 120 and Bill (b) announces that he does not have card 0:



Before the announcement Connor would have thought it possible that Bill did have card 0, i.e. $M_c 1_b$ would have been satisfied. This is no longer satisfied. Connor now knows Bill’s card ($K_c 2_b$), and also Anne’s card ($K_c 1_a$). Also, Anne now knows that Connor knows that she has card 1 ($K_a K_c 1_a$).

Chapter 5

Logic Programming

5.1 Clauses

literal	=	an atom or the negation of an atom
	=	<i>e.g.</i> p , or $\neg p$
clause	=	a disjunction of literals
	=	<i>e.g.</i> $p \vee q \vee \neg r \vee \neg s$
	=	<i>e.g. in program notation:</i> $p, q \leftarrow r, s$
Horn clause	=	a clause with at most one positive literal
	=	<i>e.g.</i> $p \vee \neg r \vee \neg s$
	=	<i>e.g. in program notation:</i> $p \leftarrow r, s$
	=	<i>(p is the head, and r, s is the tail)</i>
fact	=	a program clause with an empty tail
	=	<i>e.g.</i> $p \leftarrow$
rule	=	a program clause with a non-empty tail
	=	<i>e.g.</i> $p \leftarrow r, s$

5.2 The Martelli-Montanari unification algorithm

Set the substitution, θ to be empty.

Set the stack to contain the equation $T1 = T2$.

Set failure to *false*.

While the stack is not empty and no failure, pop $X = Y$ from the stack:

case 1 X is a variable that does not occur in Y :

replace any X s in the stack and in θ with Y s

add $X = Y$ to θ

case 2 Y is a variable that does not occur in X :

replace any Y s in the stack and in θ with X s

add $Y = X$ to θ

case 3 X and Y are identical constants or variables:

continue

case 4 X is $f(X_1, \dots, X_n)$ and Y is $f(Y_1, \dots, Y_n)$ for some functor f and $n > 0$:

push $X_i = Y_i, i = 1 \dots n$, on the stack

default failure = *true*

if failure, return failure, else return θ .

Thus, in this algorithm, if have the case where $X = f((X_1, \dots, X_n)$ but X occurs in $f((X_1, \dots, X_n)$ we fall to the default case and return failure. However, prolog does not do this and continues to try and do the unification leading to stack overflows. e.g. if $x = f(x)$, then prolog will try to unify by first giving $f(x) = f(f(x))$ then $f(f(x)) = f(f(f(x)))$ etc.

The other default case that results in failure is if $X = g(X_1, \dots, X_n)$ and $Y = f(Y_1, \dots, Y_m)$ where either $g \neq f$ or $n \neq m$.

5.3 Propositional Resolution

5.3.1 Basic Resolution Step

Given two clauses:

1. $A \leftarrow B$
2. $C \leftarrow D$

where atom a occurs in both A and C , then resolution of 1 and 2 would give:

$$A', C \leftarrow B, D'$$

where A' and D' are the results of removing some of the occurrences of a from A and D respectively.

5.3.2 Resolution Derivation or Resolution Proof

A sequence of zero or more applications of the basic resolution step using clauses in the set Π and ending in the clause ϕ . We can then write $\Pi \leftarrow \phi$.

5.3.3 Refutation

A resolution proof of the empty clause (\leftarrow).

e.g. Want to know if julia is the mother of augustus given the following information:

$Mother(julia, augustus) \leftarrow Parent(julia, augustus), Female(julia)$
 $Female(julia)$
 $Parent(julia, augustus)$

The resolution/refutation proceeds as follows:

1. $\leftarrow Mother(julia, augustus)$
2. $Mother(julia, augustus) \leftarrow Parent(julia, augustus), Female(julia)$
3. $\leftarrow Parent(julia, augustus), Female(julia)$ (resolution of 1 and 2)
4. $Female(julia)$
5. $\leftarrow Parent(julia, augustus)$ (resolution of 3 and 4)
6. $Parent(julia, augustus)$
7. \leftarrow (resolution of 5 and 6)

5.3.4 Soundness

Propositional resolution is sound. The result of resolution can always be logically entailed from clauses used. i.e. if $\Pi \vdash \phi$ then $\Pi \models \phi$.

5.3.5 Completeness

Propositional resolution is complete. If something can be entailed from a set of clauses then it can also be found with propositional resolution. However, if we only removed a single atom at a time it is incomplete. e.g.

1. $p, p \leftarrow$
2. $\leftarrow p, p$

would resolve to $p \leftarrow p$, whereas it should resolve to \leftarrow .

5.3.6 General Resolution and Search Trees

General resolution requires simultaneous substitution - see the unification algorithm.

Prolog performs a depth-first search using unification/resolution.

e.g. Given the following rules and facts:

1. $p(X, Z) \leftarrow p(X, Y), p(Y, Z)$
2. $p(Y, X) \leftarrow p(X, Y)$
3. $p(a, b) \leftarrow$

4. $p(c, b) \leftarrow$

And the goal: $\leftarrow p(a, c)$

