1. **Introduction:** ***DRAFT***

1.1 **Propositional and predicate logic**: [adapted from 5, 6, 8, 11, 12, 14]

A propositional (zeroth-order) formula is a well-formed combination of atomic propositions (usually represented by symbols such as A, B,…) and logical connectives (in this study { ¬, ∨, ∧, ←, →, ↔; `not`, `or`, `and`, `implied`, `implies`, `bi-implication`} are used). Although precedence of symbols is defined elsewhere, in this study parentheses are always used to avoid any confusion. Every formula in propositional logic takes either the value of true (⊤) or false (⊥). *Truth tables*, where every possible combination of true and false (*interpretation*) for the atoms in a formula is considered, can be used both to define the meaning of a connective and to characterize a formula. Definitions are seen in figure 1.1.

First-order (predicate) logic builds upon the foundation of propositional logic by introducing quantifiers. The universal quantifier ∀x(F) is read as '*forall x in F'*, and the existential quantifier ∃x(F) is read as there '*exists an x in F such that*'. These quantifiers allow relationships between variables (like x) to be expressed. In this study, all variables will be quantified (*bound*). If a variable is not within the scope of a quantifier, then it is *free*. (SPASS does not allow input with free variables). The terms *constants*, *variables*, *predicates*, *functions*, *relations* and *quantifiers* appear in this study, and need explanation in terms of predicate logic. The easiest way to understand these is to relate them to statements that can be interpreted in terms of real-world examples. In the statement 'Jasper is a dog', then Jasper is a constant, and dog is a predicate. This can be represented by *dog(Jasper)*, and will have a value of either true or false. The predicate dog has an arity of 1, and the constant Jasper is a predicate with arity 0. Binary predicates take two arguments (an arity of 2), and relate those arguments, for example if 'John is the owner of Jasper', then the predicate *ownerOf(John,Jasper)* has the value true. In a function the result of a statement is a unique object, for example, for *owner(Jasper)* the result is John (since Jasper is the only dog owned by John). A relation is similar to a function, except that several objects may be contained in the result, for example *inThePark(Today)* will yield many dogs including Jasper. All these formulae are ground terms since they contain no variables. If these concepts are extended from particular to general, then the argument is becomes a variable (rather that a constant), and quantifiers need to be introduced into the description. Hence if we need to encode the statement 'all animals in a given set are dogs', then this might be *∀x(dog(x))*. Likewise, 'every owner of a dog is a person' can be translated as *∀x∀y(dog(x)∧ownerOf(y,x)→person(y))*. In order to assign a truth value to to these formulae it may be necessary to assign values for the predicates, functions, and constants. Hence, *∀x(dog(x))* may be true in some *domains*, and false in others. In first order logic quantification only takes place over variables.

Figure 1.1. Definitions in propositional logic and predicate logic: (11, 12, 14, 16)

Basic syntax of propositional logic:

If F is a formula, P is an atom, then

F::= ⊤ | ⊥ | P | F | ¬F | F∨F | F∧F | F←F | F→F | F↔F	(infix notation)
F::= P | F | ¬F | ∨(F,F) | ∧(F,F) | ←(F,F) | →(F,F) | ↔(F,F)	(prefix notation)

Truth table for common logical connectives:

A	B	A∧B	A∨B	A→B
⊤	⊤	⊤	⊤	⊤
⊤	⊥	⊥	⊤	⊥
⊥	⊤	⊥	⊤	⊤
⊥	⊥	⊥	⊥	⊤

Truth table for implication:

A	B	A→B
⊤	B	B
⊥	B	⊤
A	⊤	⊤
A	⊥	¬A

Useful definitions and tautologies of propositional logic:

A∨⊤ ⫤⊨ ⊤	A∧⊤ ⫤⊨ A	A∧¬A ⫤⊨ ⊥	¬¬A ⫤⊨ A
A∨⊥ ⫤⊨ A	A∧⊥ ⫤⊨ ⊥	A∨¬A ⫤⊨ ⊤	¬⊥ ⫤⊨ ⊤
A↔B ⫤⊨ (A→B)∧(A←B)			
A→B ⫤⊨ ¬A∨B			
A∨(B∧C) ⫤⊨ (A∨B)∧(A∨C)	A∧(B∨C) ⫤⊨ (A∧B)∨(A∧C)	[Distributive Law]	
¬(A∧B) ⫤⊨ (¬A∨¬B)	¬(A∨B) ⫤⊨ (¬A∧¬B)	[De Morgan's Law]	
(A→B)∧A ⊨ B		[Inference rule, modus ponens]	

- The symbol F1 ⫤⊨ F2 is used where formulae are *logically equivalent*, meaning that, the interpretation is equal for every valuation. A, B and C are propositional literals (or modal formulae).
- For two formulae, X and F, X ⊨ F denotes that F is a *logical consequence* of X; when X is true, then F is true; any model satisfying X, also satisfies F

- Formulae created by simultaneously replacing all occurrences of all instances of any subcomponent are created by *substitution*, for example [y/x] replaces all occurrences of y by x.

Definitions in First-order Predicate Logic:

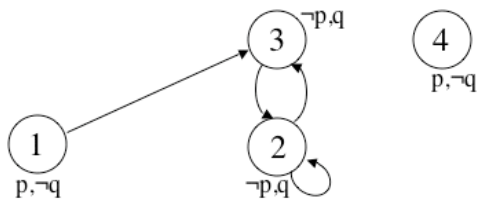
∀xF ⫤⊨ ¬∃x¬F	∀x¬F ⫤⊨ ¬∃xF	¬∀xF ⫤⊨ ∃x¬F	¬∀x¬F ⫤⊨ ∃xF

1.2 Modal Logic [adapted from 7, 9, 10]:

The syntax of the standard modal logic (propositional modal logic) considered in this study is formulated by adding two unary operators, box ☐ (or the *necessity* operator) and diamond ◇ (or the *possibility* operator) to the syntax of propositional logic. ☐ is a quantifier similar to universal quantification in predicate logic, and ◇ is similar to existential quantification. In modal logic, statements are true, false, or somewhere in-between (the are modes of truth like 'possibly false' or 'true in the future' [see figure 1.6]). Modal logic can be expressed in first-order logic, but is often easier to use modal logic to model real world concepts (that is, modal logic can form a more *expressive* language). Some of the possible meanings assigned to modal quantifiers are shown in figure 1.6. The different meanings correspond to different application domains (Note the symbols used for the modal quantifiers may change for different applications). The propositional modal logic described here can be extended to description logics, temporal logics, etc [27].

The semantics of modal logic is best visualized in terms directed graphs (di-graphs) used to illustrate Kripke semantics [24]. A Kripke frame F is an ordered pair (W, R), where W is the (non-empty) set of *possible worlds*, and R is a binary *accessibility relation* connecting some of those worlds (R={(w_1,w_2)} is taken to mean that world w_2 is accessible from world w_1). In a di-graph, the nodes represent possible worlds and the edges represent the relationships between these worlds. These figures can be taken to illustrate transitions between conceivable states (a view which is useful when model checking is applied to program verification). For example, the *frame* F = {W={1,2,3,4}, R={(1,3), (2,3), (3,2), (2,2)}} is illustrated in figure 1.2. Here for example, worlds 2 and 3 are successor worlds of world 2. In a *model* M = (W, R, I), the frame can be is given a particular *interpretation function*, for example I = {(p,{1,4}), (q,{2,3})}. The labeling of the nodes in the di-graph represents these *valuations*. Here for example, p is true in world 1 (so, 1∈I(q)), and false in world 2.

Figure 1.2 A directed graph illustrating a Kripke frame.



In Kripke semantics, the box operator (☐ϕ) takes as an argument a particular world, and returns true if the formula (ϕ) is true in *all* the direct successor worlds, of the starting world, that are defined by the accessibility relation. In a similar way, the diamond operator returns true if a particular formula is true in *at-least-one* possible successor world of the starting world. Hence, if we consider the statement M,1⊧☐☐p∨◇¬p (applied to the frame in figure 1.2), meaning that the formula ☐☐☐p∨◇¬p is true (*holds*) at world 1, then this can be verified by visual inspection: It is possible to see that ☐☐☐p is false (consider the successive transitions from worlds 1-to-3-to-2-to-2 in which p is false; likewise for successive steps 1-to-3-to-2-to-3), but that ◇¬p is true (consider the transition from worlds 1-to-3 in which p is false), and so the disjunction is true.

A more formal derivation procedure (as compared to visual inspection) has been developed for determining the truth of modal formulae. Modal formulae are subject to the *Model Checking* algorithm [7], in which the structure of the modal formula, taking account of the properties of a given frame, is expanded inductively using the transformations below.

•	M,w ⊧ p	iff	w∈I(p)	(where p is a propositional variable)
•	M,w ⊧ ⊤			
•	M,w ⊭ ⊥			
•	M,w ⊧ ¬ϕ	iff	M,w ⊭ ϕ	
•	M,w ⊧ (ϕ ∗ ψ)	iff	M,w ⊧ ϕ ∗ M,w ⊧ ψ	(where ∗ ∈ {∧ ∨ → ↔})
•	M,w ⊧ ☐ϕ	iff	∀w`∈W, (w,w`)∈R → M,w` ⊧ ϕ	
•	M,w ⊧ ◇ϕ	iff	∃w`∈W, (w,w`)∈R ∧ M,w` ⊧ ϕ	taken from [7]

For example, the following formula (in the frame in figure 1.2) is false, *M*, 1 ⊧ ☐☐p

iff ∀w∈{1, 2, 3, 4}, (1,w)∈R→*M*, w ⊧ ☐p …unfold definition ☐

iff $((1,1)\in R\rightarrow M,1 \models \Box p)\wedge((1,2)\in R\rightarrow M,2 \models \Box p)\wedge((1,3)\in R\rightarrow M,3 \models \Box p)\wedge((1,4)\in R\rightarrow M,4 \models \Box p)$ …expand quantifier
iff $(\bot\rightarrow M,1\models\Box p)\wedge(\bot\rightarrow M,2\models\Box p)\wedge(\top\rightarrow M,3\models\Box p)\wedge(\bot\rightarrow M,4\models\Box p)$ …substituting values from R
iff $\top\wedge\top\wedge(M,3\models\Box p)\wedge\top$
iff $M,3\models\Box p$
iff $(\forall w\in\{1,2,3,4\}, (3,w)\in R\rightarrow M, w \models p$ …unfold definition □
iff $(\forall w\in\{1,2,3,4\}, (3,w)\in R\rightarrow w\in I(p)$ …unfold, truth of p
iff $((3,1)\in R\rightarrow 1\in I(p)\wedge(3,2)\in R\rightarrow 2\in I(p)\wedge(3,3)\in R\rightarrow 3\in I(p)\wedge(3,4)\in R\rightarrow 4\in I(p)$ …expand quantifier
iff $((\bot\rightarrow\top)\wedge(\top\rightarrow\bot)\wedge(\bot\rightarrow\bot)\wedge(\bot\rightarrow\top)$ …substituting values from R and I
iff $(\top\wedge\bot\wedge\top\wedge\top)$
iff $\bot$

Clearly it is possible, by iteration, to consider all worlds in this interpretation, and as such proofs can be constructed regarding the truth of formulae with respect to a particular model. (Note if a formula holds in all worlds, the syntax is $M \models x$). Likewise, the generality of proofs can be further extended to cover, for example, all models (an *arbitrary world* and *arbitrary model*) in a particular frame, all models in a group of frames (the group of frames will be defined to obey a particular property), and at the most general to cover all models in all Kripke frames (properties like $\models\neg\Box\neg\phi$ iff $\Diamond\phi$ can hence be proven; for other such properties see figure 1.3).
Model checking can be extended to determine the following properties for modal formulae [7].

- *Satisfiablity*: a modal formula ϕ is satisfiable iff there is some model and some world where $M,w\models\phi$ is true.
- *Validity*: a modal formula ϕ is valid ($\models\phi$) iff it is true in all Kripke frames, for every model and every world (also known as *global satisfiablity*).
- Validity and satisfiablity are related: Often the satisfiablity of the negated formula is checked, since a modal formula ϕ is valid iff $\neg\phi$ is *unsatisfiable* (and ϕ is satisfiable iff $\neg\phi$ is not valid).

Figure 1.3: <u>Valid Propositional Modal Formulae:</u> (valid in all possible Kripke frames) [7, 9]

- All propositional tautologies
- $\models\Diamond\phi \leftrightarrow \neg\Box\neg\phi$ $\models\neg\Diamond\phi \leftrightarrow \Box\neg\phi$ $\models\Diamond\neg\phi \leftrightarrow \neg\Box\phi$
- $\models\Box\top \leftrightarrow \top$
- $\models\Diamond\bot \leftrightarrow \bot$
- $\models\Box(\phi\rightarrow\psi) \rightarrow (\Box\phi\rightarrow\Box\psi)$ or $\models(\Box(\phi\rightarrow\psi)\wedge\Box\phi) \rightarrow \Box\psi$ [*axiom K*]
- $\models\Box(\phi\wedge\psi) \leftrightarrow (\Box\phi\wedge\Box\psi)$
- $\models\Diamond(\phi\vee\psi) \leftrightarrow (\Diamond\phi\vee\Diamond\psi)$
- $\models\Diamond(\phi\wedge\psi) \rightarrow (\Diamond\phi\wedge\Diamond\psi)$
- $\models(\Box\phi\vee\Box\psi) \rightarrow \Box(\phi\vee\psi)$
- $(\models(\phi\rightarrow\psi)\wedge\models\phi)\rightarrow\models\psi$ [*necessitation*]
- $\models\phi \rightarrow \models\Box\phi$ [*modus ponens*]

In standard modal logic the two formulae *axiom K* and *necessitation* (figure 1.3) are valid for all Kripke frames. They represent the weakest restriction that can be placed on the properties of Kripke frames. Subclasses of frames can be defined by adding other constraints, which can often be represented in terms of geometric constraints (correspondence properties) on the accessibility relation (R). Some common examples are listed in figure 1.4. (Others will be seen later in this study). The properties of these Kripke frames can of course be modeled in di-graphs. For example, in figure 1.5, it can be seen that axiom D (seriality) excludes the existence of dead-end worlds (that is, worlds which are unable to see any other world). It can be demonstrated [?] that the correspondence properties listed in figure 1.4 are equivalent to modal formulae. These modal formulae are not valid in axiom K, but are however theorems of (that is, valid in) the subclass of frames restricted by the appropriate correspondence property. Hence they are known as *modal axioms*. In some cases the relationship between a modal axiom and a correspondence property can be determined automatically [10, 15]. One simple algorithm takes a modal axiom of the type $\Diamond^h\Box^i\phi\rightarrow\Box^j\Diamond^k\phi$ (where, for example, values h=0, i=1, j=2, k=3 gives an axiom $\Box\phi\rightarrow\Box\Box\Diamond\Diamond\Diamond\phi$), and defines the equivalent correspondence property as $\boldsymbol{\forall vyz\,(R^h(v,y)\wedge R^j(v,z))\rightarrow\exists x(R^i(y,x)\wedge R^k(z,x))}$ (where R^0 means equivalence($\approx$); and R^2 means composition of R(x,y) with itself, giving $\exists z(R(x,z)\wedge R(z,y))$, etc) [Scott-Lemmon algorithm, 15]. Using this algorithm, the equivalence of the modal axioms and correspondence properties listed in figure 1.4 can be demonstrated, as illustrated below:

Axiom T is $\Box\phi\rightarrow\phi$ with h=0, i=1, j=0, k=0, and correspondence property: $\forall vyz\,(R^0(v,y)\wedge R^0(v,z))\rightarrow\exists x(R^1(y,x)\wedge R^0(z,x))$
which reduces to … $((v\approx y)\wedge(v\approx z))\rightarrow\exists x(R(y,x)\wedge(z\approx x))$ and $(y\approx z)\rightarrow R(y,z)$

$\boldsymbol{\forall y\, R(y,y)}$ **Reflexive**

Axiom D is $\Box\phi\rightarrow\Diamond\phi$ with h=0, i=1, j=0, k=1, and correspondence property: $\forall vyz\,(R^0(v,y)\wedge R^0(v,z))\rightarrow\exists x(R^1(y,x)\wedge R^1(z,x))$
which reduces to … $(y\approx z)\rightarrow\exists x(R(y,x)\wedge R(z,x))$ and $\exists x(R(y,x)\wedge R(y,x))$
$\boldsymbol{\forall y\exists x(R(y,x))}$ **Serial**

Axiom B is $\Diamond\Box\phi\rightarrow\phi$ with h=1, i=1, j=0, k=0, and correspondence property: $\forall vyz\,(R^1(v,y)\wedge R^0(v,z))\rightarrow\exists x(R^1(y,x)\wedge R^0(z,x))$
which reduces to … $(R(v,y)\wedge(v\approx z))\rightarrow\exists x(R(y,x)\wedge(z\approx x))$
$\boldsymbol{\forall yz\, R(z,y)\rightarrow R(y,z)}$ **Symmetry**

Axiom 4 is $\Box\phi\rightarrow\Box\Box\phi$ with h=0, i=1, j=2, k=0, and correspondence property: $\forall vyz\,(R^0(v,y)\wedge R^2(v,z))\rightarrow\exists x(R^1(y,x)\wedge R^0(z,x))$
which reduces to … $((v\approx y)\wedge R(v,u)\wedge R(u,z))\rightarrow\exists x(R(y,x)\wedge(z\approx x))$
$\boldsymbol{\forall uyz\,(R(y,u)\wedge R(u,z))\rightarrow R(y,z)}$ **Transitive**

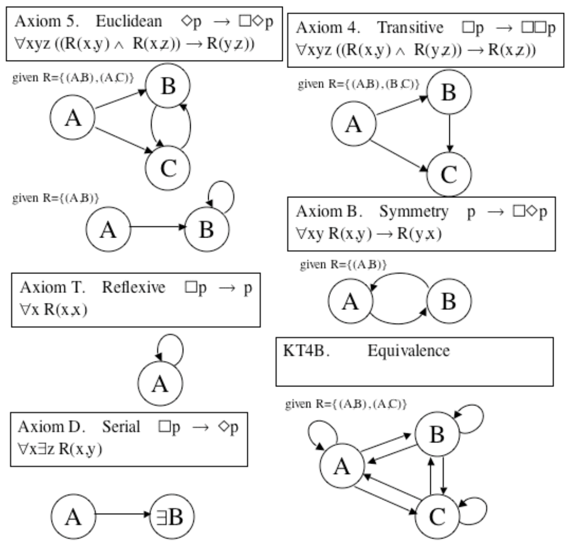
Axiom 5 is $\Diamond\Box\phi\rightarrow\Box\phi$ with h=1, i=1, j=1, k=0, and correspondence property: $\forall vyz\,(R^1(v,y)\wedge R^1(v,z))\rightarrow\exists x(R^1(y,x)\wedge R^0(z,x))$
which reduces to … $(R(v,y)\wedge R(v,z))\rightarrow\exists x(R(y,x)\wedge(z\approx x))$
$\boldsymbol{\forall vyz\,(R(v,y)\wedge R(v,z))\rightarrow R(y,z)}$ **Euclidean**

Axiom alt_1 is $\Diamond\phi\rightarrow\Box\phi$ with h=1, i=0, j=1, k=0, and correspondence property: $\forall vyz\,(R^1(v,y)\wedge R^1(v,z))\rightarrow\exists x(R^0(y,x)\wedge R^0(z,x))$
which reduces to … $(R(v,y)\wedge R(v,z))\rightarrow\exists x((y\approx x)\wedge(z\approx x))$
$\boldsymbol{\forall vyz\,(R(v,y)\wedge R(v,z))\rightarrow(y\approx z)}$ **Functional**

Figure 1.4: <u>Modal Axioms and Correspondence Properties</u>: [adapted from 1, 7, 9]

Axiom	Geometric description	Correspondence property as a restriction on Kripke frame, F = (W,R)	Correspondence property in first-order logic	Modal axiom
T	Reflexive	$\forall x\in W, (x,x)\in R$	$\forall x\, R(x,x)$	$\Box\phi\rightarrow\phi$
D	Serial	$\forall x\in W, \exists y\in W, (x,y)\in R$	$\forall x\exists y\, R(x,y)$	$\Box\phi\rightarrow\Diamond\phi$ or $\Diamond\top$
B	Symmetry	$\forall x,y\in W, ((x,y)\in R\rightarrow(y,x)\in R)$	$\forall xy(R(x,y)\rightarrow R(y,x))$	$\phi\rightarrow\Box\Diamond\phi$ or $\Diamond\Box\phi\rightarrow\phi$
4	Transitive	$\forall x,y,z\in W, (((x,y)\in R\wedge(y,z)\in R)\rightarrow(x,z)\in R)$	$\forall xyz((R(x,y)\wedge R(y,z))\rightarrow R(x,z))$	$\Box\phi\rightarrow\Box\Box\phi$
5	Euclidean	$\forall x,y,z\in W, (((x,y)\in R\wedge(y,z)\in R)\rightarrow(x,z)\in R)$	$\forall xyz((R(x,y)\wedge R(x,z))\rightarrow R(y,z))$	$\Diamond\phi\rightarrow\Box\Diamond\phi$ or $\Diamond\Box\phi\rightarrow\Box\phi$
alt_1	Functionality	$\forall x,y,z\in W, (((x,y)\in R\wedge(x,z)\in R)\rightarrow(y\approx z))$	$\forall xyz((R(x,y)\wedge R(x,z))\rightarrow(y\approx z))$	$\Diamond\phi\rightarrow\Box\phi$

Figure 1.5: <u>Examples of Kripke frames in which modal axioms hold:</u>



There are more sophisticated algorithms available [10, 15] that can generate first-order correspondence properties from other classes of target modal axiom, for example from Sahlqvist formulae, as implemented in the SCAN tool [18]. It is also important to note that there are many classes of Kripke frame restricted by a modal axiom, for which there is no first-order correspondence property (for example, McKinsey's axiom M, $\Box\Diamond\phi\rightarrow\Diamond\Box\phi$). Likewise, there are many correspondence properties do not have modal axiom equivalents (for example irreflexivity, $\forall x\in W, (x,x)\notin R$). [7]. Note also that in the general case, modal axioms are second order properties [26].

Certain well-known combinations of modal axioms, within the framework of modal logic, are used to model real world concepts. A selection is listed in figure 1.6. It is worth looking at one example more closely. In epistemic modal logic (S5), KT45 are all valid axioms (incidentally axiom D is also valid since $\models_{KT}D$), the possible worlds in the Kripke frame correspond to *knowledge states*, and the box operator $\Box\phi$ is interpreted as *knows* ϕ. Each of the individual axiom contributes a special property to this modal logic. Hence, axiom T contributes *true knowledge*, the property that everything known is true; axiom 4 contributes *positive introspection*, the property that everything known, is known to be known; axiom 5 contributes *negative introspection*, the property that everything not known, is knows to be unknown; axiom D contributes *non-contradictory knowledge*, the property that no contradictions are known; and axiom K and necessitation contribute the unimplementable property *logical omniscience*, respectively composed by the properties that, all logical consequences of knowledge are known, and all propositional tautologies are known. [7,9].

Figure 1.6: <u>Different readings of the box and diamond operator for various combinations of axioms</u>: [7,9]

Logic Description	Meaning of $\Box\phi$ and notation	Meaning of $\Diamond\phi$	axiom T	axiom 4	axiom 5	Axiom D	Axiom K
Alethic (S5)	Necessarily ϕ ($\Box\phi$)	Possibly ϕ	√	√	√	√	√
Temporal or Transitive	Always ϕ ($\Box\phi$)	Sometime in the future ϕ		√			√
Deontic or Serial	Ought to be ϕ ($\bigcirc\phi$)	Permitted to be ϕ				√	√
Doxastic KD45	Believed that ϕ ($B\phi$)	Held possible that ϕ		√	√	√	√
Epistemic KT45 or S5	Known that ϕ ($K\phi$)	Held possible that ϕ	√	√	√	√	√

1.3 <u>Translation of Modal Logic into First Order Logic</u> [adapted from 1, 7].

In order to use tools such as SPASS to attempt to prove, or to disprove, the satisfiablity of modal formulae it is necessary to first translate the modal formula into the first-order logic that can be accepted by SPASS. Such a translation has been available for some time [25, 29]. This relational translation preserves the truth state of any modal formula (and in particular the property of satisfiablity) under first order resolution (see section 1.4). The relational translation function (π below) is defined as seen below, taking as arguments the modal formula and a first-order variable symbol. Propositional variables are mapped to a unary predicate symbol (for example, p maps to Q_p), and accessibility relations are mapped to binary predicates (R is used in the uni-modal case below). The function π is applied inductively (iteratively) until no more transformations are possible.

$$\begin{aligned}
&\pi(p,x) = Q_p(x) && && \pi(\neg\phi,x) = \neg\pi(\phi,x)\\
&\pi(\top,x) = \top && && \pi(\bot,x) = \bot\\
&\pi(\phi * \psi ,x) = \pi(\phi,x) * \pi(\psi,x) && \text{where } *\in\{\to,\leftrightarrow,\vee,\wedge\} && \\
&\pi(\Box\phi,x) = \forall y(R(x,y) \to \pi(\phi,y)) && && \pi(\Diamond\phi,x) = \exists y(R(x,y) \wedge \pi(\phi,y))
\end{aligned}$$

x and y are distinct first order logic variables; ϕ and ψ are any modal formula

Modal axioms are introduced into the translation as the *correspondence properties* (restrictions on the accessibility relation), that have already been seen above (table 1.4, the 4th column entitled first order logic). These correspondence properties are incorporated into the formula to be submitted to SPASS as follows.

$$\bigwedge(\text{Correspondence Property for each Axiom}) \to \forall x\, \pi(\phi,x)$$

So for example, if a modal formula ϕ is valid in KT45, then the following is valid in first-order logic:

$$(\forall xR(x,x) \wedge \forall xyz(((R(x,y)\wedge R(x,z))\to R(y,z)) \wedge \forall xyz((R(x,y)\wedge R(x,z))\to R(y,z))) \to \forall x.\pi(\phi,x)$$

Definition of the relational translation provides the opportunity to follow the derivation of these correspondence properties. The derivations below rely upon rearranging the translation of the modal formula representing the axiom, to give a formula in which quantifiers can be eliminated by comparison to a propositional formula (provable by truth tables), and from which the correspondence properties can then be derived by substitution. These are my own derivations of these properties. A more rigorous derivation from the literature is shown below for axiom 4.

Axiom T: $\pi(\mathbf{\Box p\to p},x)$

$$\begin{aligned}
&= \forall x(\pi(\Box p,x) \to \pi(p,x))\\
&= \forall x(\forall y(R(x,y) \to \pi(p,y)) \to Q_p(x))\\
&= \forall x(\forall y(R(x,y) \to Q_p(y)) \to Q_p(x))\\
&= \forall x(\neg\forall y(R(x,y) \to Q_p(y)) \vee Q_p(x))\\
&= \forall x(\exists y(\neg(R(x,y) \to Q_p(y)) \vee Q_p(x)))\\
&= \forall x(\exists y((R(x,y) \to Q_p(y)) \to Q_p(x)))
\end{aligned}$$

$$\begin{aligned}
&= \forall x(((R(x,x) \to Q_p(x)) \to Q_p(x))) && \text{substituting } [y/x]\\
&\dashv \mathbf{\forall x(R(x,x))} && A \models (A\to B)\to B
\end{aligned}$$

proof by refutation $\neg((A\to B)\to B) \wedge A$
$\equiv \neg(\neg(\neg A\vee B)\vee B) \wedge A \equiv ((\neg A\vee B)\wedge\neg B) \wedge A \equiv ((\neg B\wedge\neg A)\vee(\neg B\wedge B)) \wedge A$
$\equiv ((\neg B\wedge\neg A)\vee(\bot)) \wedge A \equiv \neg B\wedge \neg A\wedge A \equiv \neg B\wedge \bot \equiv \bot$

Axiom D: $\pi(\mathbf{\Box p\to\Diamond p},x)$

$$\begin{aligned}
&= \forall x\, \pi(\Box p,x) \to \pi(\Diamond p,x)\\
&= \forall x(\forall y(R(x,y) \to \pi(p,y)) \to \exists z(R(x,z) \wedge \pi(p,z)))\\
&= \forall x(\forall y(R(x,y) \to Q_p(y)) \to \exists z(R(x,z) \wedge Q_p(z)))\\
&= \forall x(\exists y(R(x,y) \wedge \neg Q_p(y)) \vee \exists z(R(x,z) \wedge Q_p(z)))\\
&= \forall x\exists y\exists z((R(x,y) \to Q_p(y)) \to (R(x,z) \wedge Q_p(z)))\\
&= \mathbf{\forall x\exists y(R(x,y))} && \text{substituting } [z/y]\text{, and } (A \to B) \to (A \wedge B) \dashv\vDash A
\end{aligned}$$

proof: $(A\to B) \to (A\wedge B)$
$\equiv \neg(\neg A\vee B) \vee (A\wedge B) \equiv (A\wedge\neg B) \vee (A\wedge B) \equiv A\wedge(\neg B \vee B) \equiv A\wedge\top \equiv A$

Axiom B: $\pi(\mathbf{p\to\Box\Diamond p},x)$

$$\begin{aligned}
&= \forall x\, \pi(p,x) \to \pi(\Box\Diamond p,x)\\
&= \forall x(Q_p(x) \to \forall y(R(x,y) \to \pi(\Diamond p,y)))\\
&= \forall x(Q_p(x) \to \forall y(R(x,y) \to \exists z(R(y,z) \wedge \pi(p,z))))\\
&= \forall x(Q_p(x) \to \forall y(R(x,y) \to \exists z(R(y,z) \wedge Q_p(z))))\\
&= \forall x\forall y\exists z((R(x,y) \wedge Q_p(x)) \to (R(y,z) \wedge Q_p(z)))\\
&= \forall x\forall y((R(x,y) \wedge Q_p(x)) \to (R(y,x) \wedge Q_p(x))) && \text{substituting } [z/x]\\
&\dashv \mathbf{\forall xy\, (R(x,y) \to R(y,x))} && A\to B \vDash (A\wedge C)\to(B\wedge C)
\end{aligned}$$

proof by refutation $(A\to B) \wedge \neg((A\wedge C)\to(B\wedge C))$
$\equiv (\neg A\vee B) \wedge \neg(\neg(A\wedge C)\vee(B\wedge C)) \equiv (\neg A\vee B) \wedge ((A\wedge C)\wedge\neg(B\wedge C)) \equiv (\neg A\vee B) \wedge (A\wedge(C\wedge(\neg B\vee\neg C)))$
$\equiv (\neg A\vee B) \wedge (A\wedge((C\wedge\neg B)\vee(C\wedge\neg C))) \equiv (\neg A\vee B) \wedge (A\wedge((C\wedge\neg B)\vee\bot)) \equiv (\neg A\vee B) \wedge (A\wedge\neg B\wedge C)$
$\equiv (\neg A\vee B)\wedge\neg(\neg A\vee B) \wedge C \equiv \bot \wedge C \equiv \bot$

Axiom 4: $\pi(\mathbf{\Box p\to\Box\Box p},x)$

$$\begin{aligned}
&= \forall x\, \pi(\Box p,x) \to \pi(\Box\Box p,x)\\
&= \forall x(\forall u(R(x,u) \to \pi(p,u)) \to \forall y(R(x,y) \to \pi(\Box p,y)))\\
&= \forall x(\forall u(R(x,u) \to \pi(p,u)) \to \forall y(R(x,y) \to \forall z(R(y,z) \to \pi(p,z))))\\
&= \forall x(\forall u(R(x,u) \to Q_p(u)) \to \forall y(R(x,y) \to \forall z(R(y,z) \to Q_p(z))))\\
&= \forall x(\neg\forall u(\neg R(x,u) \vee Q_p(u)) \vee \forall y(\neg R(x,y) \vee \forall z(\neg R(y,z) \vee Q_p(z))))\\
&= \forall x(\exists u(R(x,u) \wedge \neg Q_p(u)) \vee \forall y(\neg R(x,y) \vee \forall z(\neg R(y,z) \vee Q_p(z))))\\
&= \forall x(\exists u\forall y\forall z\, ((R(x,y) \wedge R(y,z) \wedge \neg Q_p(z)) \to (R(x,u) \wedge \neg Q_p(u))))\\
&= \forall x\forall y\forall z\, ((R(x,y) \wedge R(y,z) \wedge \neg Q_p(z)) \to (R(x,z) \wedge \neg Q_p(z))) && \text{substituting } [u/z]\\
&\dashv \mathbf{\forall xyz\, ((R(x,y) \wedge R(y,z)) \to R(x,z))} && (A\wedge B)\to C \vDash (A\wedge B\wedge D)\to(C\wedge D)
\end{aligned}$$

proof by refutation $((A\wedge B)\to C) \wedge \neg((A\wedge B\wedge D)\to(C\wedge D))$
$\equiv (\neg(A\wedge B)\vee C) \wedge \neg(\neg(A\wedge B\wedge D)\vee(C\wedge D)) \equiv (\neg(A\wedge B)\vee C) \wedge ((A\wedge B\wedge D)\wedge\neg(C\wedge D))$
$\equiv (\neg(A\wedge B)\vee C) \wedge ((A\wedge B)\wedge((D\wedge\neg C)\vee(D\wedge\neg D))) \equiv (\neg(A\wedge B)\vee C) \wedge ((A\wedge B)\wedge((D\wedge\neg C)\vee\bot))$
$\equiv ((\neg(A\wedge B)\vee C)\wedge\neg C)\wedge(A\wedge B\wedge D)) \equiv ((C\wedge\neg C)\vee(\neg(A\wedge B)\wedge\neg C))\wedge(A\wedge B\wedge D)) \equiv (\neg(A\wedge B)\wedge(A\wedge B)\wedge D\wedge\neg C) \equiv \bot$

Axiom 4^κ: $\Box p \to \Box^\kappa\Box p$ is the general case. (Note axiom 4^K is equivalent to axiom 4^1).

$$\forall x(\pi(\Box p \to \Box^\kappa\Box p,x) \vDash \forall xy\, (R^{\kappa+1}(x,y) \to R(x,y))$$

Axiom 5: $\pi(\mathbf{\Diamond\Box p\to\Box p},x)$

$$\begin{aligned}
&= \forall x\, \pi(\Diamond\Box p,x) \to \pi(\Box p,x)\\
&= \forall x(\exists y(R(x,y) \wedge \pi(\Box p,y)) \to \forall z(R(x,z) \to \pi(p,z)))\\
&= \forall x(\exists y(R(x,y) \wedge \forall u(R(y,u) \to \pi(p,u))) \to \forall z(R(x,z) \to \pi(p,z))))\\
&= \forall x(\exists y(R(x,y) \wedge \forall u(R(y,u) \to Q_p(u))) \to \forall z(R(x,z) \to Q_p(z))))\\
&= \forall x(\forall y\neg(R(x,y) \wedge \forall u(\neg R(y,u) \vee Q_p(u))) \vee \forall z(\neg R(x,z) \vee Q_p(z))))\\
&= \forall x(\forall y(\neg R(x,y) \vee \exists u(R(y,u) \wedge \neg Q_p(u))) \vee \forall z(\neg R(x,z) \vee Q_p(z))))\\
&= \forall xyz\exists u(\neg R(x,y) \vee (R(y,u) \wedge \neg Q_p(u))) \vee (\neg R(x,z) \vee Q_p(z)))\\
&= \forall xyz\exists u((R(x,y) \wedge R(x,z) \wedge \neg Q_p(z)) \to (R(y,u) \wedge \neg Q_p(u))))\\
&= \forall xyz((R(x,y) \wedge R(x,z) \wedge \neg Q_p(z)) \to (R(y,z) \wedge \neg Q_p(z)))) && \text{substituting } [u/z]\\
&\dashv \mathbf{\forall xyz\, ((R(x,y) \wedge R(x,z)) \to R(y,z))} && (A\wedge B)\to C \models (A\wedge B\wedge D)\to(C\wedge D)
\end{aligned}$$

Axiom 5^κ: $\neg\Box^\kappa\neg\Box p \to \Box p$ is the general case. (Note axiom 5^κ is equivalent to axiom 5^1).

$$\forall x(\pi(\neg\Box^\kappa\neg\Box p \to \Box p,x) \vDash \qquad \forall xyz\, ((R^\kappa(x,y) \wedge R(x,z)) \to R(y,z))$$

Axiom alt$_1$: $\pi(\mathbf{\Diamond p\to\Box p},x)$

$$\begin{aligned}
&= \forall x\, \pi(\Diamond p,x) \to \pi(\Box p,x)\\
&= \forall x(\exists y(R(x,y) \wedge \pi(p,y)) \to \forall z(R(x,z) \to \pi(p,z)))\\
&= \forall x(\exists y(R(x,y) \wedge Q_p(y)) \to \forall z(R(x,z) \to Q_p(z)))\\
&= \forall x(\forall y\neg(R(x,y) \wedge Q_p(y)) \vee \forall z(\neg R(x,z) \vee Q_p(z)))\\
&= \forall x\forall y\forall z(\neg R(x,y) \vee \neg Q_p(y) \vee \neg R(x,z) \vee Q_p(z))\\
&= \forall x\forall y\forall z(\neg R(x,y) \vee \neg R(x,z) \vee \neg Q_p(y) \vee Q_p(z))\\
&= \forall x\forall y\forall z((R(x,y) \wedge R(x,z)) \to (Q_p(y) \to Q_p(z)))
\end{aligned}$$

	$\dashv$ **$\forall xyz\,((R(x,y) \wedge R(x,z)) \rightarrow (y \approx z))$**	if y equivalent to z, then statement is tautology
Axiom $alt_1^{\kappa 1,\kappa 2}$: $\neg\Box^{\kappa 1}\Box p \rightarrow \Box^{\kappa 2}\Box\neg p$	is the general case. (Note axiom $alt_1^{\kappa,\kappa}$ is equivalent to axiom $alt_1^{0,0}$).	
	$\forall x(\pi(\neg\Box^{\kappa 1}\Box p \rightarrow \Box^{\kappa 2}\Box\neg p,x) \models \forall xyz\,((R^{\kappa 1+1}(x,y) \wedge R^{\kappa 2+1}(x,z)) \rightarrow (y \approx z))$	

It is more usual (and rigorous) to derive these correspondence properties using second order quantifier elimination techniques applied to relational translation. This process has been automated in the SCAN algorithm. A single example is shown below, adapted from [26]. The translated formula is negated, transformed to clausal form, and subject to resolution (see section 1.4).

For axiom 4: $\forall p(\boldsymbol{\Box p \rightarrow \Box\Box p})$ $=$ $\forall Q_p \forall x\, \pi(\Box p \rightarrow \Box\Box p,x)$

$=$ $\forall Q_p \forall x(\forall u(R(x,u) \rightarrow Q_p(u)) \rightarrow \forall y(R(x,y) \rightarrow \forall u(R(y,z) \rightarrow Q_p(z))))$ see above

so in the negation (for resolution, see section 1.4)

$\exists Q_p \exists x\, \neg\pi(\Box p \rightarrow \Box\Box p,x)$ $=$ $\exists Q_p \exists x(\forall u(R(x,u) \rightarrow Q_p(u)) \wedge \exists y(R(x,y) \wedge \exists z(R(y,z) \wedge \neg Q_p(z))))$

and in clausal form (following Skolemization)

	1.	$\neg R(a,u) \vee Q_p(u)$	
	2.	$R(a,b)$	
	3.	$R(b,c)$	
	4.	$\neg Q_p(c)$	
applying C-resolution	5.	$\neg R(a,u) \vee c \not\approx u$	from 1, 4
	6.	$\neg R(a,c)$	from 5, by c-elimination

allowing clauses 1 and 4 to be deleted by purification, leaving clauses 2, 3, 6: $R(a,b) \wedge R(b,c) \wedge \neg R(a,c)$

relating clauses 2, 3, and 6 to the original formula, by reverse Skolemisation gives

$$\exists v_a \exists v_b \exists v_c\, [\, R(v_a,v_b) \wedge R(v_b,v_c) \wedge \neg R(v_a,v_c)\,],$$

which is then negated transitive property, so negation (to reverse the first negation) gives

$$\forall v_a \forall v_b \forall v_c [\, \neg R(v_a,v_b) \vee \neg R(v_b,v_c) \vee R(v_a,v_c)\,] \;=\; \forall v_a \forall v_b \forall v_c [\, (R(v_a,v_b) \wedge R(v_b,v_c)) \rightarrow R(v_a,v_c)\,]$$

The axiomatic translation implemented in this study builds upon the ideas presented in this translation.
Note, that there are other translation methods (for example, functional, semi-functional, and optimized-functional translations), some of which have previously been implemented in MSPASS.

1.4 <u>Resolution:</u> [adapted from 5, 7, 11, 12]

Figure 1.7 outlines the principle of resolution for propositional and then predicate formulae. SPASS is a resolution prover for first order logic based on *refutation*. In SPASS, if a formula ϕ is valid, then $\neg\phi$ is unsatisfiable (the formula is negated for proof by refutation), and the result of resolution is *Proof Found*. On the other hand, if a formula $\neg\phi$ is satisfiable, then the result of resolution is *Completion Found*. MSPASS can assess the satisfiablity of modal formulae by performing a relational translation of the modal formula into first order logic (that is then submitted to the resolution prover), and including the correspondence properties of modal axioms. Hence, a modal formula ϕ is valid in the series of logics $K\Sigma$ (where Σ represents a series of axioms) iff $\bigwedge_{A\in\Sigma}(Corr_A) \wedge \exists x.\pi(\neg\phi,x)$ is unsatisfiable in first order logic.

Many alternative proof systems exist (for example Hilbert and sequent calculi, and natural deduction, [7, 5, 12]). The only one of interest to this study is tableau, in which a tree-like diagram is created by exhaustive application of pre-formed expansion rules to the input clauses, and in which an attempt is made to close each branch of the tree that appears by derivation of an *empty clause*. If all branches are closed then the input clauses are unsatisfiable. This method is closely related to resolution. It will be noted later that under certain circumstances, the proof steps in resolution can be used to derive a tableau.

Figure 1.7: <u>Principles of Resolution:</u> (adapted from 5, 7, 11 (especially chapters 2 & 7), 12 especially chapter 6)).

<u>Some Definitions:</u>
- A formula F is *satisfiable* (true in one or more interpretations) iff $\neg F$ is *falsifiable* (false in one or more interpretations). $X \not\models F$ iff $X \wedge \neg F$ is satisfiable.
- F is *valid* (a tautology) iff $\neg F$ is *unsatisfiable* (always false). $X \models F$ iff $X \wedge \neg F$ is unsatisfiable.
- A system is *sound* if it is only possible, using the rules of the system, to draw correct conclusions regarding the above properties of any formula. It is *complete* if it is possible to prove all valid formulae with just the given rules. It is *decidable* if all calculations will terminate with a result in finite time (which may however be a very long time) for any arbitrary input.

<u>Propositional Resolution:</u>
- Resolution is a technique derived directly from modus ponens.
- A propositional formula is transformed into *conjunctive normal form* (CNF), which is defined as follows (L is literal ($P|\neg P$); P is atom; F is a disjunctive formula) $F ::= (P|\neg P)\ |\ (P|\neg P) \vee F$ and $CNF ::= F\ |\ F \wedge CNF$

  The CNF is a conjugation of disjunctions $((L_1 \vee L_2 \vee \ldots) \wedge (L_n \vee \ldots) \wedge \ldots.)$.

A propositional formula is transformed into CNF by using the following steps (see figure 1.1 for formulae):
(i) All connectives except $\wedge$, $\vee$, $\neg$ are eliminated, and $\neg$ symbols are moved inwards (using de Morgans laws)
(ii) Using the distributive laws, formulae are arranged in CNF.

An example is given below:	$A \rightarrow (B \rightarrow C) \models (A \wedge B) \rightarrow C$
negating the right-hand side for proof by refutation	$A \rightarrow (B \rightarrow C) \wedge \neg((A \wedge B) \rightarrow C)$
	$(\neg A \vee (\neg B \vee C)) \wedge \neg(\neg(A \wedge B) \vee C)$
	$(\neg A \vee \neg B \vee C) \wedge (A) \wedge (B) \wedge (\neg C)$

Note: The above inference is a tautology. In order to prove this, the procedure is repeated for $A \rightarrow (B \rightarrow C) \dashv (A \wedge B) \rightarrow C$, negating the left-hand side.

The CNF is often modified to give a stylized representation of the *clauses*, in the format $[L_1;\neg L_2]$, representing the set $\{L_1 \vee \neg L_2\}$

The example above transforms to: $[\neg A;\neg B;C][A][B][\neg C]$

- During *resolution*, *clashing clauses* are identified as complimentary literals, of the form $\boxed{L} \vee \ldots L_n$ (or C_1) and $\boxed{\neg L} \vee \ldots L_m$ (C_2), and then clashing literals are eliminated to give a *resolvent* of the form $Res(C_1,C_2) = L_n \vee L_m$.
- The resolvent clause preserves the (joint) satisfiablity state of the parent clauses, and hence resolution gradually pairs-down a set of clauses until either (i) the *empty clause* is derived indicating that the original set of clauses were *unsatisfiable*, or (ii) no further resolution steps are possible, indicating that the original set of clauses were satisfiable.

  Resolution is illustrated below using the same example: $Res([\neg A;\neg B;C], [\neg C]) = [\neg A;\neg B]$
  $Res([\neg A;\neg B], [B]) = [\neg A]$
  $Res([\neg A], [A]) = []$

  The *empty clause* "[]" is false, and unsatisfiable, so the clause set is unsatisfiable, and the original logical consequence holds.
- Propositional resolution is sound, complete and decidable.
- Conversion of formulae into CNF may give rise to an exponential increase in the number of sub-formulae, and resolution itself may produce a large number of steps, potentially making resolution a lengthy process.

  However several refinements to resolution can be made, leading to shorter (and quicker) proofs. Examples are (i) deletion of clauses containing tautologies, (ii) if clause $C_1 \subseteq C_2$, the larger clause C_2 is deleted, (iii) deletion of unit clauses, of the form [A], with concurrent deletion of any clause containing the literal A, and of the literal $\neg A$ from any other clause, and (iv) the deletion of any clause containing a literal A when the complimentary literal $\neg A$ is not found anywhere in the clause set. Each of the refinements, at a minimum, preserves the satisfiablity of the smaller clause set that is submitted to resolution. [All from 6]

<u>Predicate (first-order) Resolution:</u>
- Predicate resolution follows the rules above, but has the additional complications arising from quantified variables.
- After the CNF is formed, a *prenex* format is generated in which all quantifiers are extracted from the body of the formula. All bound variables are renamed apart, and quantifiers are then extracted by applying the transformations $Qx(A) * B \dashv\models Qx(A * B)$, where $Q=\forall|\exists$ and $*=\vee|\wedge$ and x is not free in B (For an explanation see chapter 8 of [12], chapter 9 of [5]). When possible existential quantifiers are extracted first, since this makes Skolemisation easier (see next).
- Existential quantifiers are removed by *Skolemisation*. An $\exists x$ at the head (leftmost) of the list of quantifiers is removed, and bound x-variables in the body of the formula are replaced by a unique *Skolem constant*. Where $\exists x$ appears behind a list of universal quantifiers, the bound x-variables in the body are replaced by a unique function in each of the universally quantified variables (for example, $\forall y \forall z \exists x$ raises the replacement of x by f(y,z); an excellent explanation is given on chapter 7 of [11]. Note that Skolemisation preserves satisfiablity, but not logic equivalence). Finally all the universal quantifiers are dropped.
- Clashing clauses are resolved by substitution of variables. A *most general unifier* is formulated that describes a set of simultaneous substitutions that allows sub-formulae involving variables to be unified [see chapter 7 of 11 for a excellent explanation].
- Predicate resolution is only complete if another technique, such as positive *factoring* is included. In factoring, duplicate copies of a term are eliminated from any clause to remove repetition.
- Predicate resolution is **not a decision procedure** (and hence may never terminate, that is never produce a result), since for even very simple clause sets there may be an infinite number of unique substitutions possible (for example the clause set $[\neg p(x);p(f(x))][p(a)]$ from [6]).

  A simple example of resolution illustrates these principles.

$\forall x(P(x) \rightarrow (Q(x) \rightarrow R(x))) \models \forall x((P(x) \wedge Q(x)) \rightarrow R(x))$	
$\forall x(P(x) \rightarrow (Q(x) \rightarrow R(x))) \wedge \neg(\forall y\, ((P(y) \wedge Q(y)) \rightarrow R(y)))$	renaming apart / negation for refutation
$\forall x(\neg P(x) \vee \neg Q(x) \vee R(x)) \wedge \exists y(P(y) \wedge Q(y) \wedge \neg R(y))$	
$\exists y(\forall x(\neg P(x) \vee \neg Q(x) \vee R(x)) \wedge (P(y) \wedge Q(y) \wedge \neg R(y)))$	
$\exists y \forall x((\neg P(x) \vee \neg Q(x) \vee R(x)) \wedge (P(y) \wedge Q(y) \wedge \neg R(y)))$	
$\forall x((\neg P(x) \vee \neg Q(x) \vee R(x)) \wedge (P(c) \wedge Q(c) \wedge \neg R(c)))$	c is a Skolem constant
$[\neg P(x);\neg Q(x);R(x)]\ [P(c)]\ [Q(c)]\ [\neg R(c)]$	clause set
$Res([\neg P(c);\neg Q(c);R(c)]\ [P(c)]) = [\neg Q(c);R(c)]$	substituting [x/c]
$Res([\neg Q(c);R(c)]\ [Q(c)]) = [R(c)]$	
$Res([R(c)]\ [\neg Q(c);R(c)]\ [\neg R(c)]) \;=\; []$	unsatisfiable, so inference holds

- Again, a large number of improvements to the basic resolution are possible, which improve efficiency, and may be capable of avoiding non-terminating calculations. Many of these techniques eliminate unnecessary resolution steps, and hence reduce the space over which to search for a solution. See [28] for a comprehensive review. In terms of this study ordered resolution with selection and hyper-resolution are most relevant.

1.5. Essential Knowledge of SPASS. [adapted from 17, 18]

There are a large variety of input modes for SPASS, but the discussion here is restricted to those appropriate for this study. A selection of other input features may be accessed in the examples provided in the web-based interface developed for this project at the button *more info* under the menu tab *sample scripts (Sc)* (see `http://www2.cs.man.ac.uk/~smithk/KJSpass/main.html`)*)*. The anatomy of a .dfg file is illustrated by the example below. Both problems and program control options may be input from a .dfg file. The location of a .dfg file is input from the command line as … *SPASS <path>example.dfg*.

Figure 1.8: A .dfg input for SPASS and the results output by SPASS:

INPUT .dfg file	OUTPUT of SPASS
1 begin_problem(example).	1 Input Problem:
% NO line numbers in .dfg	2 1[0:Inp] || -> P(skc1)*.
	3 2[0:Inp] || -> Q(skc1)*.
2 **list_of_descriptions.**	4 3[0:Inp] || R(skc1)* -> .
3 name({**}).	5 4[0:Inp] || Q(U) P(U) -> R(U)*.
4 author({**}).	6 This is a monadic Horn problem without equality.
5 status(unknown).	7 This is a problem that has, if any, a finite domain model.
6 description({**}).	8 There are no function symbols.
7 end_of_list.	9 This is a problem that contains sort information.
	10 The conjecture is ground.
	11 The following monadic predicates have finite extensions: Q, P.
8 **list_of_symbols.**	12 Axiom clauses: 1 Conjecture clauses: 3
9 predicates[(Q,1),(P,1),(R,1)].	13 Inferences: IEmS=1 ISoR=1 IORe=1
10 end_of_list.	14 Reductions: RFMRR=1 RBMRR=1 RObv=1 RUnC=1 RTaut=1 RSST=1 RSSi=1
	15 RFSub=1 RBSub=1 RCon=1
	16 Extras : Input Saturation, Always Selection, No Splitting, Full
11 **list_of_formulae(axioms).**	17 Reduction, Ratio: 5, FuncWeight: 1, VarWeight: 1
12 formula(	18 Precedence: div > id > R > Q > P > skc0 > skc1
13 forall([x],implies(P(x),	19 Ordering : KBO
14 implies(Q(x),R(x))))	20 Processed Problem:
15).	21 Worked Off Clauses:
16 end_of_list.	22 Usable Clauses:
	23 1[0:Inp] || -> P(skc1)*.
	24 2[0:Inp] || -> Q(skc1)*.
17 **list_of_formulae(conjectures).**	25 3[0:Inp] || R(skc1)* -> .
18 formula(	26 6[0:Res:1.0,4.1] || Q(skc1) -> R(skc1)*.
18 forall([x],	27 7[0:Res:4.2,3.0] || P(skc1) Q(skc1)* -> .
20 implies(and(P(x),Q(x)),R(x)))	28 4[0:Inp] || P(U) Q(U) -> R(U)*.
21).	29 SPASS V 3.0
22 end_of_list.	30 **SPASS beiseite: Proof found.**
	31 Problem: /tmp/mspass13564.in
	32 SPASS derived 3 clauses, backtracked 0 clauses and kept 5 clauses.
23 **list_of_settings(SPASS).**	33 SPASS allocated 575 KBytes.
24 {*	34 **SPASS spent 0:00:00.03 on the problem.**
25 set_flag(DocProof,1).	35 0:00:00.00 for the input.
26 set_flag(PProblem,1).	36 0:00:00.00 for the FLOTTER CNF translation.
27 set_flag(PGiven,1).	37 0:00:00.00 for inferences.
28 set_flag(Sorts,0).	38 0:00:00.00 for the backtracking.
29 set_flag(Splits,0).	39 0:00:00.00 for the reduction.
30 set_flag(CNFRenaming,0).	40 Here is a proof with depth 1, length 6 :
31 set_flag(CNFOptSkolem,0).	41 1[0:Inp] || -> P(skc1)*.
32 set_flag(CNFStrSkolem,0).	42 2[0:Inp] || -> Q(skc1)*.
33 *}	43 3[0:Inp] || R(skc1)* -> .
34 end_of_list.	44 4[0:Inp] || P(U) Q(U) -> R(U)*.
	45 5[0:Res:2.0,4.0] || P(skc1) -> R(skc1)*.
35 end_problem.	46 8[0:MRR:5.0,5.1,1.0,3.0] || -> .
	47 Formulae used in the proof : conjecture0 axiom0

Key features are described below:

- The problem name appears in line 1. It may only contain alphanumeric characters.
- The sections can include (i) Description (lines 2-7), (ii) Symbols (lines 8-10), (iii) Axiom formulae (lines 11-16), (iv) Conjecture formulae (lines 17-22), (v) Settings (lines 23-34).
- **Description** list: All items are self explanatory except for status. Status may take values `satisfiable`, `unsatisfiable` and `unknown`, referring to the expected outcome of the problem. All other fields may contain any character, and are often left blank, but are not optional.
- **Symbols** list: Defines symbols occurring in the subsequent first order and modal formulae. Contents of interest to this study are

(i) *predicates*: used to define non-standard symbols used in the formulae sections, in the format `(symbol, arity)`

(ii) *translpairs*: (not shown in the example) Used to map a first-order predicate symbols to a modal logic propositional symbol. In this study a typical use is, for example, `translpairs[(R,r)]` with predicates[(R,2),(r,0)…]., linking the modality index `r` from modal formulae, to `R` in a correspondence property defined in the dfg file.

- **Formulae** (for first order logic problems) or **prop_formulae** (for modal problems, converted by the software into first order logic problems; see figure ?? for the typical syntax) may occur in either the **axioms** or **conjectures** lists. Each of the axioms and conjectures lists may contain multiple formulae. The axioms (A) and conjectures (C) lists are connected as follows: $\neg((A_1 \wedge A_2 \wedge \ldots A_n) \rightarrow (C_1 \vee C_2 \vee \ldots C_n))$, so conjectures are automatically negated. Formulae are entered in prefix notation, and for this study, the operators used in first order formulae are `and`, `or`, `not`, `implies`, `implied`, `equiv`, `equal`, `true`, `false`, with quantifiers `forall` and `exists` (requiring two arguments; a list of terms, and the quantified sub-formulae), and in modal prop_formulae the quantifiers are `box` and `dia`,
- **Settings** list: flags are set which control command line options (see brief descriptions at `http://spass.mpi-inf.mpg.de/webspass/help/options.html`.

Many options can also be set at the command-line: *SPASS –AnOption=value filename.dfg*.

The precedence of symbols defined in the symbols list can be defined with the syntax `set_precedence(a,B,z,C)`.

Typical options used to run SPASS are set in the dfg file above and can be seen in figure ??. Some options important for this study are described below [17, 18]. In most cases, setting the flag value to 1 enables the option, and 0 disables the option.

-Auto=1 enables automatic configuration (of inference and reduction rules, sort and ordering technology, precedence, and the splitting and selection strategies chosen) yielding a complete calculation. (By default enabled).

Several options are available to increase the information printed, and are useful for debugging, etc.

-DocProof=1 prints details of the (i) proof found or (ii) the saturated set of clauses in a failed proof. Storing these clauses slows calculations significantly, and should be avoided when measuring execution times. (lines 40-46).

Other options that may be useful are:

-PProblem=1 prints the input clause set.

-PGiven=1 prints the input clause selected to perform inferences.

-PKept=1 prints non-redundant clauses generated by inferences, but not clauses derived by reduction and saturation.

-PFlags=1 prints the set value of all flags. (Not used in the example above).

Several advanced resolution technologies are available in SPASS. They are by default enabled, but for this study should be disabled.

-Sorts=0 disables sort constraints on negative literals.

-CNFOptSkolem=0 disables optimized Skolemization.

-CNFStrSkolem=0 disables strong Skolemization.

-CNFRenaming=0 disables formula renaming.

Since first order resolution is not decidable, it is wise to bound the resources allocated to any problem.

TimeLimit=200 limits the execution time for proof search to approximately 200 seconds. This limit may be greatly exceeded, because the cutoff is only checked when a new clause is selected for inferences (Default=–1 means unlimited execution time).

An example of the output produced in seen in figure 1.8, and illustrates some features of SPASS output.

- The input problem is reported in a modified clausal form (lines 1-5), which correspond to $\top \rightarrow P(c)$, $\top \rightarrow Q(c)$, $R(c) \rightarrow \bot$, $(Q(x) \wedge P(x)) \rightarrow Q(c)$. The maximal literal is marked with *.
- The resolution proof is reported in lines 40-46, with the clauses used listed first, followed by the resolution steps (here line 45 is Res(2,4)=[¬P(c); R(c)]), and the final line (46) describing the derivation of the empty clause.
- SPASS is a refutation-based resolution prover for first order logic. The final result (line 30) may contain (i) `SPASS beiseite: Proof found` if the input formulae have a model and are valid (`unsatisfiable`), (ii) `SPASS beiseite: Completion found` if the input formula is not valid (`satisfiable`), and since (iii) validity is not a decidable problem, the calculation may run forever, or be terminated without result by a time limit.
- The output describes various technologies (inferences, simplifications, reduction rules) chosen for use during the attempted proof, for example lines 13-19, These are of minimal interest for this study.
- The total CPU-time used is reported as seen on line `34,` and further breakdowns of execution times by module (for example the eml-module) may be reported (lines 34-39).