AUTOMATED REASONING

SLIDES 3-6 Proofs and Things (Appendix 1)

NOTE: ALL OPTIONAL MATERIAL

Proof of Soundness of Resolution
Proof of Skolemisation Theorem
Proof of “subfree” subsumption property

KB - AR - 13

Some Useful Proofs
A1ai

The slides Appendix1 (A1) contain various proofs about resolution. The theorems in A1b and A1c are important as they give the basis for the soundness of the resolution principle. The Skolemisation theorem on A1ci means that it is sound to consider the clausal form representation of a problem, rather than the general first order representation when using refutation as a proof technique to show (un)satisfiability. (This was called (**) on Optional Slide 4di.) The theorem on A1bi means that when proving theorems about resolution id is allowed to restrict them to Herbrand interpretations and models as opposed to arbitrary models and interpretations, which is usually much easier. (This was called Useful theorem (*) on Slide 4bii.) There is also a proof of the property Subfree, mentioned in Optional part of Slides 6.

Some of the information on unifiers should be familiar to you from Prolog. But notice that Prolog does not test for the *occurs check* condition: the check, for equation xi=ti, that xi is not in ti. This is done for efficiency, but it can lead to unsoundness (of Prolog). The traditional counterexample to this unsoundness is succeeding to show that $\forall x\exists y P(x,y) \models \exists y \forall x P(x,y)$ (which is *incorrect*). The (Skolemised) clausal form of the Data+negated conclusion (i.e of $\forall x\exists y P(x,y)$ and $\forall y\exists x\neg P(x,y)$) comprises the two clauses P(x,f(x)) and ¬P(g(y),y). (Remember that each ∃ quantifier must give rise to different Skolem functions.) These two literals do not unify as the occurs check fails. The unification algorithm first gives x=g(y) and f(x)=y, and then x==g(y) and f(g(y))=y, but the latter fails the occurs check. However, if you try the Prolog query P(g(y),y), with the data P(x,f(x)) it succeeds. If you try to write the answer out - well, try it!

Soundness of a single Resolution step
A1bi

Recall from Slides 4 that the Soundness proof of resolution requires only to consider Herbrand models and to show that clauses $S \models_H R(C1,C2)$, where C1 and C2 are in S and R(C1,C2) is their resolvent. i.e. if M is an H-model of S then M is an H-model of S+R(C1,C2). (Note that R(C1,C2) does not introduce any terms not already occurring in the language of S.)

Theorem (Single step Soundness): Let $C1 = \forall[G \vee H]$, $C2 = \forall[\neg E \vee F]$, $R = \forall[(H \vee F)\theta]$ and $G\theta = E\theta$ and $mgu(G,E) = \theta$. (Here, G and E are atoms, F and H are clauses and the ∀ indicates universal quantification over variables in the clause.) Then,

if M is a H-model of $\forall[G \vee H]$ and $\forall[\neg E \vee F]$, then M is a H-model of $\forall[(H \vee F)\theta]$

Proof:
- Variables in C1 and C2 can be renamed so that C1 and C2 are "standardised apart" (i.e. have no variables in common).
- The implicit universal quantifiers can be drawn out into a prefix to yield

$\forall[C1 \wedge C2] \models_H \forall[C1\theta \wedge C2\theta]$ (**)

$\equiv \forall[(G \vee H)\theta \wedge (\neg E \vee F)\theta] \equiv \forall[(\neg H \to G)\theta \wedge (E \to F)\theta]$

$\equiv \forall[(\neg H\theta \to G\theta) \wedge (E\theta \to F\theta)] \models \forall[(\neg H\theta \to F\theta)] \equiv \forall[(H \vee F)\theta]$

The step (**) is the crucial one. It says that if M is a H-model of $\forall[C1 \wedge C2]$ then M is also a H-model of $\forall[C1\theta \wedge C2\theta]$. This follows from the fact that if θ is the mgu of the step then it only uses terms in Sig(C1,C2), though it may use variables too. (DIY! An outline proof is shown on A1bii.)

Note that the contrapositive of Single Step Soundness states that if $\forall[C1\theta \wedge C2\theta]$ has no H-model then $\forall[C1 \wedge C2]$ has no H-model.

Soundness of a single Resolution step (continued)
A1bii

Theorem: Let θ be a substitution for variables in C using terms constructed from a given signature Σ and possibly some new variables. Then $\forall C \models_H \forall C\theta$.

Proof: To show that $\forall C\theta$ is true, show that $C\theta$ holds for every substitution of ground terms from Σ to the variables in $C\theta$. Let ρ be such a substitution. Without loss of generality, let the variables in C be x1, ..., xn and θ is the substitution {xi==ti}, where the ti may be the same as xi in case no substitution is made by θ for xi. Otherwise, each ti is either a ground term in Σ or a term involving new variables. Let the variables in $C\theta$ be yi, ..., ym and ρ = {yi==gi}, where gi are ground terms using Σ. Then $C(\theta\rho) = (C\theta)\rho$ by definition of composition of substitutions (See Slide 3aviii). Hence θρ is a ground substitution of C and since ∀C is true by the premise, C(θρ) is true also. Therefore, since ρ is arbitrary, (Cθ)ρ is true.

The Single step soundness theorem suffices to show Resolution Soundness which also relies on the following *Useful Theorem* (*)

Useful Theorem (*)
Corresponding to any model M of S there is a Herbrand model HM of S.
or equivalently, If S has no Herbrand models then S has no models.
So when showing S has no models, it is sufficient to show S has no H-models.

(**Note also**: If S has no models it clearly has no Hmodels, so with the above theorem we have the property that S has no models iff S has no Hmodels.)

(The idea of the proof of this theorem is on A1biii. You can find more details if interested in Chapter 1 of notes at www/doc.ic.ac.uk/~kb)

Soundness of a single Resolution step (continued)

A1biii

The proof idea of (*) is given next.

Let S be a set of clauses using signature Σ. Starting from a model I of S construct HM, a H-model of S as follows. Each atom in the HB is assigned a truth value in HM given by

P(t1, ..., tn) = I(P)(I(t1), ..., I(tn)).

Let C be a clause in C and suppose C is assigned true by I. Then for each substitution of domain values for variables in C, some literal in C is assigned true by I. Without loss of generality, consider one substitution σ for C and suppose that after the substitution I makes literal L in C true. In case literal L is positive in C (and = P(g1, ..., gn)), then by the definition of HM above the assignment in HM makes true all atoms P(t1, ..., tn) such that for all i, I(ti) = gi. These atoms will be ground instances of L. (Similar considerations apply for negative L, or if C is false in I). C is true in HM, since the substitution σ was arbitrary.

Theorem: (Soundness of Resolution) If S⇒*[] then S has no models
The proof uses induction on the length of the refutation of S.
Base Case: k=0. S must contain the empty clause and is clearly unsatisfiable.
Case k>0. Assume as induction hypothesis (IH) that for refutations of length k-1 if S⇒*[] then S has no H-models. Such a refutation has the form (for some C1 and C2 in S) S ⇒ S+R(C1,C2) ⇒* []; i.e. after the initial step none or more resolution steps lead to the empty clause. By (IH) (S+R(C1,C2)) has no H-models ==> S has no H-models (by the contrapositive of Single step soundness) ==> S has no models (by Useful Theorem *)

It is not difficult to extend the Soundness of Resolution proof to include factoring.
i.e. to show that $S \models_H F$, where C is in S and F is a factor of C, and show that if S ⇒ *[ ] by derivations using resolution and factoring then S has no models.

Skolemisation Theorem

A1ci

The Skolemisation part of conversion to clausal form can be implemented by the function Sk1 below. Then we can show (see also below) that
∀V Sk1(E,V) has a model iff ∀V E has a model, for free variables V in E. **(*)**

Skolem(A) = Sk1(A,∅)
Sk1(A,V)= A, if A is a literal
Sk1(A op B,V) = Sk1(A,V) op Sk1(B,V), where "op" is ∧ / ∨
Sk1(∀x.A, V) = ∀x.Sk1(A,V ∪{x})
Sk1(∃x.A,V) = ∃x.Sk1(A[x/f(V')],V),
where f is a unique function,V⊇V', V' occur in A
No other cases are necessary as negations are adjacent to atoms.

Required to show: Skolem(E) has a model iff E has a model.
Since E is a sentence it has no free variables and the property (*) will yield the result immediately. We prove the property (*) by induction on the structure of E.
Assume as Induction Hypothesis that property (*) holds for immediate subterms of E.
Next we show the property holds for E.

Case E is a literal:
M is a model of ∀V .Sk1(E,V) iff M is a model of ∀V.E (defn. of Sk1)

Case E is A op B:
M is a model of ∀V .Sk1(A op B,V)
iff M is a model of ∀V [Sk1(A,V) op Sk1(B,V)] (defn. of Sk1)
iff M is a model of ∀V [Sk1(A,V)] 'op' M is a model of ∀V [Sk1(B,V)
iff M is a model of ∀V A 'op' M is a model of ∀V B (Ind. Hyp.)
iff M is a model of ∀V [A op B]

A1cii

Case E is ∀x .A:
M is a model of ∀V.Sk1(∀x.A,V) iff M is a model of ∀V,x.Sk1(A,V∪{x}) (defn. Sk1)
iff M is a model of ∀V,x.A (Ind. Hyp.) iff M is a model of ∀V.(∀x. A) (Equiv.)

Case E is ∃x. A:
M is a model of ∀V.Sk1(∃x.A,V) iff M is a model of ∀V.Sk1(A[x/(f(V')],V) (defn. Sk1)
iff M is a model of ∀V.A[x/f(V')] (Ind. Hyp.) iff M is a model of ∀V.∃x.A (below)

The very last step in the case for *∃x. A* is the one that does the Skolemisation and it is proved next. The notation x/f(V') means x is replaced by f(V'):

Suppose M is a model of ∀V.∃x .A. To give a model for ∀V. A[x/f(V')], we need to extend M so it includes an interpretation for f.

For each vector D', of elements from the domain of M, ∃x .A[V'/D',x] is true (since ∀V.∃x .A), so interpret f by : f(D') = some z: A[V'/D', x/z] is true.
Then A[V'/D', x/f[D')] is true in M and M is a model of ∀V. A[x/f(V')]

Suppose now that M is a model of ∀V. A[x/f(V')].
Then for each vector D' of elements from the domain of M, A[V'/D', x/f(D')] is true.
Hence ∃x .A [V'/D'] is true and so ∀V∃x .A is true too.

The details of the other parts are easier and are left as an exercise.

A1di

About Subsumption:

Slides 6 discussed how using subsumed clauses leads to redundancy in a proof and informally introduced the Property Subfree:

Property SubFree: Let S be a set of unsatisfiable clauses. Then, there is a refutation Ref from S such that for each clause Ck at depth k≥0 and used in Ref to derive a clause at depth >k, Ck is not subsumed by any different clause derived at depth ≤k.

In other words, no resolvent in the refutation R is subsumed by a clause in S or by a previously generated clause

The proof of Property SubFree uses this fact (illustrated on slides 6biv/bv):
if C subsumes D and a step in a refutation uses D (resolving with K) to derive R,
then either C subsumes R,
or resolving C and K leads to resolvent R' that subsumes R.
The proof of this fact is not difficult and is left as an exercise.

Here we show that the *Property SubFree* holds for refutations formed using saturation search. The proof uses the notion of *maximum depth of a refutation*, which is the stage in the generation of resolvents in a refutation by saturation search at which the empty clause is formed. A resolvent R is *derived in a refutation at depth k* if k is the stage in the saturation search at which R is derived. A given clause is *derived at depth 0*. The inductive proof constructs a subsumption free refutation in stages from an arbitrary refutation.

Throughout this section assume that any factoring is combined with the resolution step that uses the factor. e.g. if L1∨L2∨C is resolved with ¬L3∨D, where L1,L2,L3 unify with mgu θ and C and D are clauses, then the resolvent is (C∨D)θ, as if first is made a factor step between L1 and L2 and then a resolution step using ¬L3. This simplifies the proof by avoiding the restriction to strict subsumption. . Also assume that by subsumption is always meant θsubsumption.

Proof of Property SubFree: Let S be a set of unsatisfiable clauses and let Ref be some refutation starting from clauses S with maximum depth m. If Ref already possesses Property SubFree there is nothing to prove. Otherwise, let the last violation of Property SubFree occur in Ref at depth n≥1. The proof uses induction on n.

Case n=m. Some clause at depth n (=m) is the empty clause, formed by resolving two facts D1 and D2. If a clause C subsumes D1 then C will resolve with D2 also to form the empty clause. Similarly if C subsumes D2. Hence D1 (D2) can be removed from its use to derive [ ] at depth m of the refutation. Ref will then possess Property SubFree as there are no more violations in Ref.

Induction step. (n<m). Assume as Induction Hypothesis (IH) that for any refutation of clauses from S of maximum depth ≤m and such that the last violation of Property SubFree occurs at depth k, where k<n, a refutation Ref satisfying Property SubFree can be found. Let R1 at depth n<m be derived from clauses D1 and D2 such that a clause C subsumes D1, where D1, D2 and C are all derived at depth < n. Assume that if several clauses subsume D1, C is selected to be minimal in the subsumption order amongst these clauses.

A1dii

A1diii

Continued from A1dii

By the aforementioned fact, either C subsumes R1 or C resolves with D1 to form R1’ which subsumes R1. A new refutation is constructed from Ref as follows.

Clauses in Ref at depth <n remain the same. Clause R1 is replaced by C if C subsumes R1, otherwise it is replaced by the resolvent R1’ of C with D1. In both cases the replacement clause subsumes R1. In steps at depths >n, a similar replacement is made using the new subsuming clause C, until either depth m is reached, the empty clause is formed at depth <m, or the new resolvent is unchanged from the resolvent in Ref. There are a finite number of such replacements as the maximum depth m cannot be increased by using subsumed clauses, it can only be decreased. In effect, these replacements allow for new subsumptions by R1’ to be propagated through the remainder of the refutation Ref’.

After repeating such replacements as necessary for all clauses derived at depth n, the resulting refutation Ref’ will have maximal depth ≤m (in case [ ] was reached earlier than at depth m), possibly with some duplicated clauses. Moreover, the last violation of Property SubFree, if any, is now at depth <n, due to the minimality of C (you should try to show this). Hence by the induction hypothesis a refutation can be found from Ref’ that does not violate Property SubFree. In applying the hypothesis, some clauses may be made redundant (if they are no longer used), and duplicated clauses are finally removed.

(Exercise: If interested in proofs, you might like to construct a (simple) example of a refutation that violates the Property and then to follow the construction to obtain a refutation that does satisfy it.)