The Equality Axioms 12bii

Reasoning with equality in resolution and in tableau implicitly makes use of a set of clausal axiom schema and the reflexivity of equality (EQAX1). There are 2 basic substitutivity schema:

(i) those that deal with substitution at the argument level of atoms (EQAX3), and
(ii) those that deal with substitution at the argument level of terms (EQAX2).
They are given on Slide 12biii.

An alternative form of EQAX combines the schema for each argument place into a single schema that will deal with one or more arguments at the same time. They are:

EQAX2 (Alternative) ∀[x1=y1∧...∧ xn=yn → **f(x1,…,xn)=f(y1,…,yn)]**
EQAX3 (Alternative) ∀[x1=y1∧...∧ xn=yn ∧ **P(x1,…,xn) → P(y1,…,yn)]**

Exercise (a jolly good one!): Show that the two forms of EQAX are equivalent.
Hint: To show EQAX2(Alternative) implies EQAX2 (and similarly for EQAX3) is easy. You need to use Reflexivity. The other direction is a bit harder.

A discussion of models and interpretations of Equality is given later.

Equality Axioms

12biii

Reasoning with equality "naturally" uses the *equality axioms* implicitly
EQAX1 ∀x[x=x]
EQAX2 ∀[xi=yi → f(x1,…,xi,…,xn)=f(x1,…,yi,…,xn)]
EQAX3 ∀[xi=yi ∧ P(x1,…,xi,…,xn) → P(x1,…,yi,…,xn)]

EQAX2 and EQAX3 as clauses:
EQAX2 ∀[¬xi=yi ∨ f(x1,…,xi,…,xn)=f(x1,…,yi,…,xn)]
EQAX3 ∀[¬xi=yi ∨ ¬P(x1,…,xi,…,xn) ∨ P(x1,…,yi,…,xn)]

EQAX2 and EQAX3 are *substitutivity* schema.
There is one axiom for each argument position for each function/predicate.

There is an equivalent form of the Equality Axioms, which are also useful.

Alternative form for EQAX2 and EQAX3:
EQAX2 (Alternative)
∀[¬x1=y1 ∨...∨ ¬xn=yn ∨ **f(x1,…,xn) = f(y1,…,yn)]**
EQAX3 (Alternative)
∀[¬x1=y1 ∨...∨ ¬xn=yn ∨ **¬P(x1,…,xn) ∨ P(y1,…,yn)]**

RUE-RESOLUTION (Digricoli,Raptis) (Uses alternative form of EQAX)

EQAX2 (Alt) ∀[¬x1=y1 ∨...∨ ¬xn=yn ∨ **f(x1,…,xn) = f(y1,…,yn)]**
EQAX3 (Alt) ∀[¬x1=y1 ∨...∨ ¬xn=yn ∨ **¬L(x1,…,xn) ∨ L(y1,…,yn)]**

Given C1≡L(t1,...,tn) ∨ D and C2≡¬L'(t1',...,tn') ∨ E
the RUE-resolvent is D ∨ E ∨ ¬t1=t1' ∨ .... ∨ ¬tn=tn'
where, in EQAX3,
L(t1,...,tn) unifies with L(x1,...,xn) and L'(t1',...,tn') unifies with L(y1,...,yn)

RUE forces a kind of *locking* on use of alternative EQAX
The locking gives ¬x1=y1, ..., ¬xn=yn higher indices than other literals

Informal example:
P(a)∨D, ¬P(b) and ¬x=y ∨ ¬P(x) ∨ P(y) (ie C1, C2 and EQAX3) ==> D ∨ ¬a=b

To match P(a) and ¬ P(b) (to resolve C1 and C2) must show a = b.
The goal "show a=b" is represented by ¬a=b
and it is refuted **after** matching P(a),P(b)

12civ

Notes on Hyper-paramodulation 12cvi

The simulation of Hyper-paramodulation using Hyper-resolution (HR) and equality axioms shows soundness of paramodulation. For completeness, we'd like to show that a hyper-paramodulation refutation can be constructed from a HR refutation using also EQAX.

Suppose there is a HR refutation using EQAX. Then
Use of EQAX3 simulates a paramodulation step already
Use of EQAX2 can also be turned into a paramodulation step using reflexive axioms such as f(x)=f(x). **(Details an exercise.)**

Notes on RUE-resolution
RUE-resolution is an alternative to paramodulation as a way of including EQAX implicitly into the deduction. It can, informally, be interpreted as trying to impose locking onto the use of equality axioms. It is as though some kind of locking strategy is applied to EQAX3 such that the non-equality literals must be resolved (with other clauses) before any other useful resolvents can be made using these axioms. i.e. the equality literals are locked highest in EQAX3. The alternative form of EQAX3 (and EQAX2) are the most appropriate to use here. That is:

EQAX2 (Alternative) ∀[x1=y1∧...∧ xn=yn → **f(x1,…,xn)=f(y1,…,yn)]**
EQAX3 (Alternative) ∀[x1=y1∧...∧ xn=yn ∧ **P(x1,…,xn) → P(y1,…,yn)]**

Thus the basic step is to match two potentially complementary literals with the two "P" literals in the appropriate EQAX3 schema. The result is a disjunction of inequalities, which can then be resolved with either EQAX1, EQAX2, or equations in the data.