BFO

By bittner

September 7, 2008

Contents

theory *EMR*

imports *FOL*

begin
typedecl *Rg*

arities *Rg* :: *term*

consts

OR :: *Rg* => *Rg* => *o*
POR :: *Rg* => *Rg* => *o*
PR :: *Rg* => *Rg* => *o*
PPR :: *Rg* => *Rg* => *o*
Sum :: *Rg* => *Rg* => *Rg* => *o*
Diff :: *Rg* => *Rg* => *Rg* => *o*
Prod :: *Rg* => *Rg* => *Rg* => *o*
UniR :: *Rg* => *o*

axioms

PR-refl: (*ALL a. PR*(*a*,*a*))
PR-antisym: (*ALL a b.* (*PR*(*a*,*b*) & *PR*(*b*,*a*) --> *a*=*b*))
PR-trans: (*ALL a b c.* (*PR*(*a*,*b*) & *PR*(*b*,*c*) --> *PR*(*a*,*c*)))
PR-diff: (*ALL a b.* (~*PR*(*a*,*b*)) --> (*EX c. Diff*(*a*,*b*,*c*)))
PR-sum: (*ALL a b.* (*EX c. Sum*(*a*,*b*,*c*)))
PR-prod: (*ALL a b.* (*OR*(*a*,*b*) --> (*EX c. Prod*(*a*,*b*,*c*))))

defs

OR-def: *OR*(*a*,*b*) == (*EX c.* (*PR*(*c*,*a*) & *PR*(*c*,*b*)))
POR-def: *POR*(*a*,*b*) == *OR*(*a*,*b*) & ~*PR*(*a*,*b*) & ~*PR*(*b*,*a*)

PPR-def: *PPR(a,b) == PR(a,b) & ~PR(b,a)*
Sum-def: *Sum(a,b,c) == (ALL d. OR(d,c) <−> (OR(d,a) | OR(d,b)))*
Diff-def: *Diff(a,b,c) == (ALL d. OR(d,c) <−> (EX e. (PR(e,a) & ~OR(e,b) & OR(e,d))))*
Prod-def: *Prod(a,b,c) == (ALL d. OR(d,c) <−> (OR(d,a) & OR(d,b)))*
UniR-def: *UniR(a) == (ALL b. PR(b,a))*

lemma *PR-refl-rule*: *PR(a,a)*
⟨*proof*⟩

lemma *PR-trans-rule*: [|*PR(a,b)*; *PR(b,c)*|] ==> *PR(a,c)*
⟨*proof*⟩

lemma *PR-antisym-rule*: [|*PR(a,b)*; *PR(b,a)*|] ==> *a=b*
⟨*proof*⟩

lemma *PR-diff-rule*: ~*PR(a,b)*==>(*EX c. Diff(a,b,c)*)
⟨*proof*⟩

lemma *PPR-imp-PR*: *PPR(a,b)* ==> *PR(a,b)*
⟨*proof*⟩

theorem *PPR-asym*: *PPR(a,b)* ==> ~*PPR(b,a)*
⟨*proof*⟩

theorem *PPR-trans*: [|*PPR(a,b)*;*PPR(b,c)*|]==>*PPR(a,c)*
⟨*proof*⟩

theorem *PR-imp-PPR-or-Id*: *PR(a,b)* ==> (*PPR(a,b)* | (*a=b*))
⟨*proof*⟩

theorem *PPR-or-Id-imp-PR*: (*PPR(a,b)* | (*a=b*)) ==> *PR(a,b)*
⟨*proof*⟩

theorem *PR-and-PPR-imp-PPR*: [|*PR(a,b)*;*PPR(b,c)*|] ==> *PPR(a,c)*
⟨*proof*⟩

theorem *PPR-and-PR-imp-PPR*: [|*PPR(a,b)*;*PR(b,c)*|] ==> *PPR(a,c)*
⟨*proof*⟩

theorem *OR-refl*: *OR(a,a)*
⟨*proof*⟩

theorem *OR-sym*: *OR(a,b) ==> OR(b,a)*
⟨*proof*⟩

theorem *POR-sym*: *POR(a,b) ==> POR(b,a)*
⟨*proof*⟩

theorem *POR-irrefl*: ~*POR(a,a)*
⟨*proof*⟩

theorem *PR-imp-OR*: *PR(a,b) ==> OR(a,b)*
⟨*proof*⟩

theorem *PR-and-OR*: [|*PR(a,b)*;*OR(a,c)*|] ==> *OR(b,c)*
⟨*proof*⟩

theorem *PR-and-notOR-imp-notOR*: [|*PR(a,b)*;~*OR(b,c)*|] ==> ~*OR(a,c)*
⟨*proof*⟩

theorem *notOR-sym*: ~*OR(a,b)* ==> ~*OR(b,a)*
⟨*proof*⟩

theorem *PR-ssuppl*: ~*PR(a,b)* ==> (*EX c*. (*PR(c,a)* & ~*OR(c,b)*))
⟨*proof*⟩

lemma *PR-ssuppl-transpos*: ~(*EX c*. (*PR(c,a)* & ~*OR(c,b)*)) ==> *PR(a,b)*
⟨*proof*⟩

theorem *OR-imp-OR-imp-PR*: (*ALL c*. (*OR(c,a)* --> *OR(c,b)*)) ==> *PR(a,b)*
⟨*proof*⟩

theorem *OR-ident* : (*ALL a b*. (*ALL c*. (*OR(c,a)* <-> *OR(c,b)*)) <-> *a=b*)
⟨*proof*⟩

theorem *Sum-unique*: [|*Sum(a,b,c)*;*Sum(a,b,d)*|] ==> *c = d*
⟨*proof*⟩

theorem *Sum-imp-PR-and-PR*: *Sum(a,b,c)* ==> *PR(a,c)* & *PR(b,c)*
⟨*proof*⟩

theorem *Sum-sym*: *Sum(a,b,c)* ==> *Sum(b,a,c)*
⟨*proof*⟩

theorem *Sum-refl*: *Sum*(*a*,*a*,*a*)
⟨*proof*⟩

theorem *Sum-imp-id*: *Sum*(*a*,*a*,*b*) ==> *a*=*b*
⟨*proof*⟩

theorem *PR-imp-Sum*: *PR*(*a*,*b*) ==> *Sum*(*a*,*b*,*b*)
⟨*proof*⟩

theorem *Diff-unique*: [|*Diff*(*a*,*b*,*c*);*Diff*(*a*,*b*,*d*)|] ==> *c* = *d*
⟨*proof*⟩

theorem *Prod-unique*: [|*Prod*(*a*,*b*,*c*);*Prod*(*a*,*b*,*d*)|] ==> *c* = *d*
⟨*proof*⟩

theorem *Diff-imp-notPR*: *Diff*(*a*,*b*,*c*) ==> ~*PR*(*a*,*b*)
⟨*proof*⟩

theorem *notOR-imp-Diff*: ~*OR*(*a*,*b*) ==> *Diff*(*a*,*b*,*a*)
⟨*proof*⟩

theorem *Diff-imp-PR*: *Diff*(*a*,*b*,*c*) ==> *PR*(*c*,*a*)
⟨*proof*⟩

theorem *PR-and-notPR-imp-notPR*: [|*PR*(*a*,*b*);~*PR*(*c*,*b*)|] ==> ~*PR*(*c*,*a*)
⟨*proof*⟩

theorem *PR-and-Diff-impl-Diff-PR*: [|*PR*(*a*,*b*);*Diff*(*c*,*b*,*d*)|] ==> (*EX e*. (*Diff*(*c*,*a*,*e*) & *PR*(*d*,*e*)))
⟨*proof*⟩
thm *conjI*
⟨*proof*⟩

theorem *UniR-unique*: [|*UniR*(*a*);*UniR*(*b*)|] ==> *a*=*b*
⟨*proof*⟩

end

theory *QSizeR*

imports *EMR*

begin

consts

SSR :: *Rg* => *Rg* => *o*
Sym :: *Rg* => *Rg* => *Rg* => *o*
LER :: *Rg* => *Rg* => *o*

axioms

SSR-refl: *ALL a. SSR(a,a)*
SSR-sym: *SSR(a,b)* ==> *SSR(b,a)*
SSR-trans: [|*SSR(a,b)*;*SSR(b,c)*|] ==> *SSR(a,c)*
PR-and-SSR-imp-PR: [|*PR(a,b)*;*SSR(a,b)*|] ==> *PR(b,a)*
SSR-plus: [|*Plus(a,c,d1)*;*Plus(b,c,d2)*;*Sym(c,a,b)*|] ==> (*SSR(a,b)* <−> *SSR(d1,d2)*)
LER-total: *ALL a b.* (*LER(a,b)* | *LER(b,a)*)
LER-and-LER-imp-SSR: [|*LER(a,b)*;*LER(b,a)*|] ==> *SSR(a,b)*

lemma *SSR-refl-rule*: *SSR(a,a)*
⟨*proof*⟩

defs

Sym-def: *Sym(c,a,b)* == (*ALL d.* (*PR(d,c)* −−> (*PR(d,a)* <−> *PR(d,b)*)))
LER-def: *LER(a,b)* == (*EX c.* (*SSR(c,a)* & *PR(c,b)*))

consts

RSSR :: *Rg* => *Rg* => *o*
NEGR :: *Rg* => *Rg* => *o*
SSCR :: *Rg* => *Rg* => *o*
LRSSR :: *Rg* => *Rg* => *o*
LNRSSR :: *Rg* => *Rg* => *o*

defs

NEGR-def: *NEGR(a,b)* == (*EX c1 c2.* (*SSR(c1,a)* & *PR(c1,b)* & *Diff(b,c1,c2)*
& *RSSR(b,c2)*)))
SSCR-def: *SSCR(a,b)* == (~*NEGR(a,b)* & ~*NEGR(b,a)*)
LRSSR-def: *LRSSR(a,b)* == (*LER(a,b)* | *RSSR(a,b)*)
LNRSSR-def: *LNRSSR(a,b)* == (*LRSSR(a,b)* & ~*RSSR(a,b)*)

axioms

RSSR-refl: *ALL a. RSSR(a,a)*
RSSR-sym: *RSSR(a,b)* ==> *RSSR(b,a)*
RSSR-between: [|*RSSR(a,b)*;*LER(a,c)*;*LER(c,b)*|] ==> (*RSSR(c,a)* & *RSSR(c,b)*)

RSSR-and-NEGR-imp-NEGR: [|*RSSR*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
NEGR-and-RSSR-imp-NEGR: [|*NEGR*(*a*,*b*);*RSSR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)

NEGR-and-LER-imp-NEGR: [|*NEGR*(*a*,*b*);*LER*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)

RSSR-sum: [|*Sum*(*a*,*c*,*d1*);*Sum*(*b*,*c*,*d2*);*Sym*(*c*,*a*,*b*);*RSSR*(*a*,*b*)|] ==> *RSSR*(*d1*,*d2*)
RSSR-sum2: [|*Sum*(*a*,*b*,*c*);*NEGR*(*a*,*c*)|] ==> ~*LRSSR*(*b*,*a*)

theorem *Id-imp-SSR*: *a*=*b* ==> *SSR*(*a*,*b*)
⟨*proof*⟩

theorem *PR-and-PR-imp-SSR*: [|*PR*(*a*,*b*);*PR*(*b*,*a*)|] ==> *SSR*(*a*,*b*)
⟨*proof*⟩

theorem *PR-and-SSR-imp-Id*: [|*PR*(*a*,*b*);*SSR*(*a*,*b*)|] ==> *a* = *b*
⟨*proof*⟩

theorem *PR-imp-LER*: *PR*(*a*,*b*) ==> *LER*(*a*,*b*)
⟨*proof*⟩

theorem *PPR-imp-notSSR*: *PPR*(*a*,*b*) ==> ~*SSR*(*a*,*b*)
⟨*proof*⟩

theorem *LER-refl*: *LER*(*a*,*a*)
⟨*proof*⟩

theorem *LER-and-SSR-imp-LER*: [|*LER*(*a*,*b*);*SSR*(*b*,*c*)|] ==> *LER*(*a*,*c*)
⟨*proof*⟩

theorem *SSR-and-LER-imp-LER*: [|*SSR*(*c*,*a*);*LER*(*a*,*b*)|] ==> *LER*(*c*,*b*)
⟨*proof*⟩

theorem *SSR-imp-LER*: *SSR*(*a*,*b*) ==> *LER*(*a*,*b*)
⟨*proof*⟩

theorem *SSR-imp-LER-and-LER*: *SSR*(*a*,*b*) ==> (*LER*(*a*,*b*) & *LER*(*b*,*a*))
⟨*proof*⟩

theorem *LER-trans*: [|*LER*(*a*,*b*);*LER*(*b*,*c*)|] ==> *LER*(*a*,*c*)
⟨*proof*⟩

thm *exE*
⟨*proof*⟩
thm *exI*
⟨*proof*⟩

theorem *SSR-and-RSSR-imp-RSSR*: [|*SSR*(*a*,*b*);*RSSR*(*b*,*c*)|] ==> *RSSR*(*a*,*c*)
⟨*proof*⟩
thm *SSR-and-LER-imp-LER*
⟨*proof*⟩
thm *RSSR-between*
⟨*proof*⟩

theorem *RSSR-and-SSR-imp-RSSR*: [|*RSSR*(*a*,*b*);*SSR*(*b*,*c*)|] ==> *RSSR*(*a*,*c*)
⟨*proof*⟩

theorem *SSR-imp-RSSR*: *SSR*(*a*,*b*) ==> *RSSR*(*a*,*b*)
⟨*proof*⟩

theorem *NEGR-imp-LER*: *NEGR*(*a*,*b*) ==> *LER*(*a*,*b*)
⟨*proof*⟩

theorem *NEGR-imp-LER-and-notSSR*: *NEGR*(*a*,*b*) ==> (*LER*(*a*,*b*) & ~*SSR*(*a*,*b*))
⟨*proof*⟩

theorem *NEGR-irrefl*: *ALL a*. (~*NEGR*(*a*,*a*))
⟨*proof*⟩

theorem *NEGR-assym*: *NEGR*(*a*,*b*) ==> ~*NEGR*(*b*,*a*)
⟨*proof*⟩

theorem *LER-and-NEGR-imp-NEGR*: [|*LER*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *SSR-and-NEGR-imp-NEGR*: [|*SSR*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *NEGR-and-SSR-imp-NEGR*: [|*NEGR*(*a*,*b*);*SSR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *NEGR-trans*: [|*NEGR*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *PR-and-NEGR-imp-NEGR*: [|*PR*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *NEGR-and-PR-imp-NEGR*: [|*NEGR*(*a*,*b*);*PR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *RSSR-imp-notNEGR*: *RSSR*(*a*,*b*) ==> ~*NEGR*(*a*,*b*)
⟨*proof*⟩

theorem *SSCR-refl*: *ALL a*. *SSCR*(*a*,*a*)
⟨*proof*⟩

theorem *SSCR-sym*: *SSCR*(*a*,*b*) ==> *SSCR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-refl*: *LRSSR*(*a*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-LRSSR-imp-RSSR*: [|*LRSSR*(*a*,*b*);*LRSSR*(*b*,*a*)|] ==> *RSSR*(*a*,*b*)
⟨*proof*⟩

theorem *RSSR-imp-LRSSR-and-LRSSR*: *RSSR*(*a*,*b*) ==> (*LRSSR*(*a*,*b*) & *LRSSR*(*b*,*a*))
⟨*proof*⟩

theorem *RSSR-iff-LRSSR-and-LRSSR*: *RSSR*(*a*,*b*) <-> (*LRSSR*(*a*,*b*) & *LRSSR*(*b*,*a*))
⟨*proof*⟩

theorem *LRSSR-and-NEGR-imp-NEGR*: [|*LRSSR*(*a*,*b*);*NEGR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *NEGR-and-LRSSR-imp-NEGR*: [|*NEGR*(*a*,*b*);*LRSSR*(*b*,*c*)|] ==> *NEGR*(*a*,*c*)
⟨*proof*⟩

theorem *LRSSR-total*: *LRSSR*(*a*,*b*) | *LRSSR*(*b*,*a*)
⟨*proof*⟩

theorem *LNRSSR-asym*: *LNRSSR*(*a*,*b*) ==> ~*LNRSSR*(*b*,*a*)
⟨*proof*⟩

theorem *LNRSSR-trans*: [|*LNRSSR*(*a*,*b*);*LNRSSR*(*b*,*c*)|] ==> *LNRSSR*(*a*,*c*)
⟨*proof*⟩

end
theory *RBG*

imports *EMR QSizeR*

begin

consts

SpR :: *Rg* => *o*
MxSpR :: *Rg* => *Rg* => *Rg* => *o*
CoPPR :: *Rg* => *Rg* => *o*
CGR :: *Rg* => *Rg* => *o*
CNGSpR :: *Rg* => *Rg* => *o*
ECR :: *Rg* => *Rg* => *o*
DCR :: *Rg* => *Rg* => *o*

defs

MxSpR-def: MxSpR(a,b,c) == SpR(a) & SpR(b) & SpR(c) & PR(a,c) & PR(b,c) & ~OR(a,b) & (ALL e. (SpR(e) & PR(a,e) --> (a = e | OR(e,b) | ~PR(e,c))))
CoPPR-def: CoPPR(a,b) == SpR(a) & SpR(b) & PPR(a,b) & (ALL c d. (MxSpR(c,a,b) & MxSpR(d,a,b) --> SSR(c,d)))
CGR-def: CGR(a,b) == (EX c. (SpR(c) & OR(c,a) & OR(c,b) & (ALL d. (CoPPR(d,c) --> (OR(d,a) & OR(d,b))))))
CNGSpR-def: CNGSpR(a,b) == SpR(a) & SpR(b) & (EX ca cb. (CoPPR(ca,cb) & MxSpR(a,ca,cb) & MxSpR(b,ca,cb)))
ECR-def: ECR(a,b) == CGR(a,b) & ~OR(a,b)
DCR-def: DCR(a,b) == ~CGR(a,b)

axioms

SP-nested: [|Sp(a);Sp(b);Sp(c);MxSpR(u,a,c);MxSpR(v,a,c);(ALL ua va. (((MxSpR(ua,a,b) & MxSpR(va,a,b)) | (MxSpR(ua,b,c) & MxSpR(va,b,c))) --> SSR(ua,va)))|] ==> SSR(u,v)
SP-PPR-exists: EX b. (SpR(b) & PPR(b,a))

SSR-imp-CoPPR-and-MxSpR: [|SpR(a);SpR(b);SSR(a,b)|] ==> (EX ca cb. (CoPPR(ca,cb) & MxSpR(a,ca,cb) & MxSpR(b,ca,cb)))
CGR-imp-CGR-imp-PR: (ALL c. (CGR(c,a) --> CGR(c,b))) ==> PR(a,b)

SpR-CoPPR-NEGR-exists: SpR(a) ==> (EX b. CoPPR(b,a) & NEGR(b,a))
SpR-and-LER-and-notSSR-imp-CoPPR-and-SSR-exists: [|SpR(a);LER(b,a);~SSR(b,a)|] ==> (EX c. (CoPPR(c,a) & SSR(c,b)))

SpR-and-SpR-and-PP-imp-MaxSpR: [|SpR(a);SpR(b);PPR(a,b)|] ==> (EX c. (MxSpR(c,a,b)))

lemma *SP-PR-exists*: *EX b. (SpR(b) & PR(b,a))*
⟨*proof*⟩

lemma *CoPPR-imp-PPR*: *CoPPR(a,b) ==> PPR(a,b)*
⟨*proof*⟩

lemma *CoPPR-imp-SpR-and-SpR*: *CoPPR(a,b) ==> (SpR(a) & SpR(b))*
⟨*proof*⟩

theorem *CoPPR-asym*: *CoPPR(a,b) ==> ~CoPPR(b,a)*
⟨*proof*⟩

theorem *CoPPR-trans*: *[|CoPPR(a,b);CoPPR(b,c)|] ==> CoPPR(a,c)*
⟨*proof*⟩

theorem *PPR-exists*: *(EX b. PPR(b,a))*
⟨*proof*⟩

theorem *Sp-CoPPR-exists*: *SpR(a) ==> (EX b. CoPPR(b,a))*
⟨*proof*⟩

theorem *PR-and-NEGR-exists*: *EX b. (PR(b,a) & NEGR(b,a))*
⟨*proof*⟩

theorem *CGR-refl*: *CGR(a,a)*
⟨*proof*⟩

theorem *CGR-sym*: *CGR(a,b) ==> CGR(b,a)*
⟨*proof*⟩

theorem *PR-imp-CGR-imp-CGR*: *PR(a,b) ==> (ALL c. (CGR(c,a) --> CGR(c,b)))*
⟨*proof*⟩

theorem *OR-imp-CGR*: *OR(a,b) ==> CGR(a,b)*
⟨*proof*⟩

theorem *PR-iff-CGR-imp-CGR*: *PR(a,b) <-> (ALL c. (CGR(c,a) --> CGR(c,b)))*
⟨*proof*⟩

theorem *Id-iff-CGR-iff-CGR*: *a=b <-> (ALL c. (CGR(c,a) <-> CGR(c,b)))*

⟨*proof*⟩

theorem *PR-imp-CGR*: *PR(a,b)* ==> *CGR(a,b)*
⟨*proof*⟩

theorem *PR-and-CGR-imp-CGR*: [|*PR(a,b)*;*CGR(a,c)*|] ==> *CGR(b,c)*
⟨*proof*⟩

theorem *PR-and-notCGR-imp-notCGR*: [|*PR(a,b)*;~*CGR(b,c)*|] ==> ~*CGR(a,c)*
⟨*proof*⟩

theorem *Sp-sum*: *OR(w,a)* <−> (*EX b*. (*SpR(b)* & *PR(b,a)* & *OR(w,b)*))
⟨*proof*⟩

theorem *Sp-imp-CNGSpR*: *SpR(a)* ==> *CNGSpR(a,a)*
⟨*proof*⟩

theorem *CNGSpR-imp-SSR*: *CNGSpR(a,b)* ==> *SSR(a,b)*
⟨*proof*⟩

theorem *Sp-and-SSR-imp-CNGSpR*: [|*SpR(a)*;*SpR(b)*;*SSR(a,b)*|] ==> *CNGSpR(a,b)*
⟨*proof*⟩

theorem *Sp-imp-SSR-iff-CONSpR*: [|*SpR(a)*;*SpR(b)*|] ==> (*SSR(a,b)* <−> *CNGSpR(a,b)*)
⟨*proof*⟩

theorem *CNGSpR-imp-Sp-and-Sp*: *CNGSpR(a,b)* ==> (*SpR(a)* & *SpR(b)*)
⟨*proof*⟩

theorem *CNGSpR-sym*: *CNGSpR(a,b)* ==> *CNGSpR(b,a)*
⟨*proof*⟩

theorem *CNGSpR-trans*: [|*CNGSpR(a,b)*;*CNGSpR(b,c)*|] ==> *CNGSpR(a,c)*
⟨*proof*⟩

theorem *ECR-sym*: *ECR(a,b)* ==> *ECR(b,a)*
⟨*proof*⟩

theorem *ECR-irrefl*: ~*ECR(a,a)*
⟨*proof*⟩

theorem *DCR-sym*: *DCR(a,b)* ==> *DCR(b,a)*
⟨*proof*⟩

theorem *DCR-irrefl*: ~*DCR*(*a*,*a*)
⟨*proof*⟩

end
theory *QDiaSizeR*

imports *RBG*

begin

consts

SSRdia :: *Rg* => *Rg* => *o*
LERdia :: *Rg* => *Rg* => *o*
MinBSpR :: *Rg* => *Rg* => *o*

BR :: *Rg* => *Rg* => *Rg* => *o*

defs

MinBSpR-def: *MinBSpR*(*a*,*b*) == *SpR*(*a*) & *PR*(*b*,*a*) & (*ALL c*. (*SpR*(*c*) & *PR*(*b*,*c*) --> *LER*(*a*,*c*)))
SSRdia-def: *SSRdia*(*a*,*b*) == (*EX ca cb*. (*MinBSpR*(*ca*,*a*) & *MinBSpR*(*cb*,*b*) & *SSR*(*ca*,*cb*)))
LERdia-def: *LERdia*(*a*,*b*) == (*EX ca cb*. (*MinBSpR*(*ca*,*a*) & *MinBSpR*(*cb*,*b*) & *LER*(*ca*,*cb*)))

BR-def: *BR*(*a*,*b*,*c*) == *SpR*(*a*) & *SpR*(*b*) & *SpR*(*c*) & (*EX sab sbc sac bab bbc bac*. (*Sum*(*a*,*b*,*sab*) & *Sum*(*b*,*c*,*sbc*) & *Sum*(*a*,*c*,*sac*) & *MinBSpR*(*bab*,*sab*) & *MinBSpR*(*bbc*,*sbc*) & *MinBSpR*(*bac*,*sac*) & *PR*(*bab*,*bac*) & *PR*(*bbc*,*bac*)))

axioms

MinBSpR-exists: (*EX b*. *MinBSpR*(*b*,*a*))
PR-and-MinBSpR-and-MinBSpR-imp-PR: [|*PR*(*a*,*b*);*MinBSpR*(*aa*,*a*);*MinBSpR*(*bb*,*b*)|] ==> *PR*(*aa*,*bb*)

BR-trans: [|*BR*(*a*,*b*,*w*);*BR*(*b*,*c*,*w*)|] ==> *BR*(*a*,*b*,*c*)
BR-connect: [|*BR*(*a*,*b*,*w*);*BR*(*a*,*c*,*w*)|] ==> (*BR*(*a*,*b*,*c*) | *BR*(*a*,*c*,*b*))

theorem *MinBSpR-imp-PR*: *MinBSpR*(*a*,*b*) ==> *PR*(*b*,*a*)
⟨*proof*⟩

theorem *MinBSpR-and-MinBSpR-imp-SSR*: [|MinBSpR(a,c);MinBSpR(b,c)|] ==> SSR(a,b)
⟨*proof*⟩

theorem *MinBSpR-unique*: [|MinBSpR(a,c);MinBSpR(b,c)|] ==> a=b
⟨*proof*⟩

theorem *SpR-iff-MinBSpR*: SpR(a) <-> MinBSpR(a,a)
⟨*proof*⟩

theorem *EX-SpR-CGR-and-CGR*: (EX c. (SpR(c) & CGR(c,a) & CGR(c,b)))
⟨*proof*⟩

theorem *BR-imp-PR*: BR(a,b,a) ==> PR(b,a)
⟨*proof*⟩

theorem *BR-refl*: [|SpR(a);SpR(b)|] ==> BR(a,a,b)
⟨*proof*⟩

theorem *BR-sym*: BR(a,b,c) ==> BR(c,b,a)
⟨*proof*⟩

theorem *SSRdia-refl*: SSRdia(a,a)
⟨*proof*⟩

theorem *SSRdia-sym*: SSRdia(a,b) ==> SSRdia(b,a)
⟨*proof*⟩

theorem *SSRdia-trans*: [|SSRdia(a,b);SSRdia(b,c)|] ==> SSRdia(a,c)
⟨*proof*⟩

theorem *LERdia-refl*: LERdia(a,a)
⟨*proof*⟩

theorem *LERdia-trans*: [|LERdia(a,b);LERdia(b,c)|] ==> LERdia(a,c)
⟨*proof*⟩

theorem *LERdia-and-LERdia-imp-SSRdia*: [|LERdia(a,b);LERdia(b,a)|] ==> SS-

Rdia(*a*,*b*)
⟨*proof*⟩

theorem *SSRdia-imp-LERdia-and-LERdia*: *SSRdia*(*a*,*b*) ==> (*LERdia*(*a*,*b*) & *LERdia*(*b*,*a*))
⟨*proof*⟩

theorem *SSRdia-iff-LERdia-and-LERdia*: *SSRdia*(*a*,*b*) <−> (*LERdia*(*a*,*b*) & *LERdia*(*b*,*a*))
⟨*proof*⟩

theorem *LERdia-or-LERdia*: (*LERdia*(*a*,*b*) | *LERdia*(*b*,*a*))
⟨*proof*⟩

theorem *Id-imp-SSRdia*: *a*=*b* ==> *SSRdia*(*a*,*b*)
⟨*proof*⟩

theorem *PR-and-PR-imp-SSRdia*: [|*PR*(*a*,*b*);*PR*(*b*,*a*)|] ==> *SSRdia*(*a*,*b*)
⟨*proof*⟩

theorem *PR-imp-LERdia*: *PR*(*a*,*b*) ==> *LERdia*(*a*,*b*)
⟨*proof*⟩

theorem *LERdia-and-SSRdia-imp-LERdia*: [|*LERdia*(*a*,*b*);*SSRdia*(*b*,*c*)|] ==> *LERdia*(*a*,*c*)
⟨*proof*⟩

theorem *SSRdia-and-LERdia-imp-LERdia*: [|*SSRdia*(*a*,*b*);*LERdia*(*b*,*c*)|] ==> *LERdia*(*a*,*c*)
⟨*proof*⟩

theorem *SpR-and-SpR-and-SSR-imp-SSRdia*: [|*SpR*(*a*);*SpR*(*b*);*SSR*(*a*,*b*)|] ==> *SSRdia*(*a*,*b*)
⟨*proof*⟩

theorem *SpR-and-SpR-and-SSRdia-imp-SSR*: [|*SpR*(*a*);*SpR*(*b*);*SSRdia*(*a*,*b*)|] ==> *SSR*(*a*,*b*)
⟨*proof*⟩

theorem *SpR-and-SpR-imp-SSR-iff-SSRdia*: [|*SpR*(*a*);*SpR*(*b*)|] ==> (*SSR*(*a*,*b*) <−> *SSRdia*(*a*,*b*))
⟨*proof*⟩

theorem *SSRdia-imp-LERdia*: *SSRdia*(*a*,*b*) ==> *LERdia*(*a*,*b*)
⟨*proof*⟩

consts

RSSRdia :: *Rg* => *Rg* => *o*
NEGRdia :: *Rg* => *Rg* => *o*

defs

RSSRdia-def: *RSSRdia*(*a*,*b*) == (*EX ca cb*. (*MinBSpR*(*ca*,*a*) & *MinBSpR*(*cb*,*b*) & *RSSR*(*ca*,*cb*)))
NEGRdia-def: *NEGRdia*(*a*,*b*) == (*EX ca cb*. (*MinBSpR*(*ca*,*a*) & *MinBSpR*(*cb*,*b*) & *NEGR*(*ca*,*cb*)))

theorem *RSSRdia-refl*: *RSSRdia*(*a*,*a*)
⟨*proof*⟩

theorem *RSSRdia-sym*: *RSSRdia*(*a*,*b*) ==> *RSSRdia*(*b*,*a*)
⟨*proof*⟩

theorem *RSSRdia-between*: [|*RSSRdia*(*a*,*b*);*LERdia*(*a*,*c*);*LERdia*(*c*,*b*)|] ==> (*RSSRdia*(*c*,*a*) & *RSSRdia*(*c*,*b*))
⟨*proof*⟩

theorem *SSRdia-and-RSSRdia-imp-RSSRdia*: [|*SSRdia*(*a*,*b*);*RSSRdia*(*b*,*c*)|] ==> *RSSRdia*(*a*,*c*)
⟨*proof*⟩

theorem *RSSRdia-and-SSRdia-imp-RSSRdia*: [|*RSSRdia*(*a*,*b*);*SSRdia*(*b*,*c*)|] ==> *RSSRdia*(*a*,*c*)
⟨*proof*⟩

theorem *SSRdia-imp-RSSRdia*: *SSRdia*(*a*,*b*) ==> *RSSRdia*(*a*,*b*)
⟨*proof*⟩

theorem *NEGRdia-imp-LERdia*: *NEGRdia*(*a*,*b*) ==> *LERdia*(*a*,*b*)
⟨*proof*⟩

theorem *NEGRdia-imp-LERdia-and-notSSRdia*: *NEGRdia*(*a*,*b*) ==> (*LERdia*(*a*,*b*) & ~*SSRdia*(*a*,*b*))
⟨*proof*⟩

theorem *NEGRdia-irrefl*: ~*NEGRdia*(*a*,*a*)
⟨*proof*⟩

theorem *NEGRdia-assym*: *NEGRdia(a,b)* ==> ~*NEGRdia(b,a)*
⟨*proof*⟩

theorem *LERdia-and-NEGRdia-imp-NEGRdia*: [|*LERdia(a,b)*;*NEGRdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *NEGRdia-and-LERdia-imp-NEGRdia*: [|*NEGRdia(a,b)*;*LERdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *SSRdia-and-NEGRdia-imp-NEGRdia*: [|*SSRdia(a,b)*;*NEGRdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *NEGRdia-SSRdia-imp-NEGRdia*: [|*NEGRdia(a,b)*;*SSRdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *NEGRdia-trans*: [|*NEGRdia(a,b)*;*NEGRdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *PR-and-NEGRdia-imp-NEGRdia*: [|*PR(a,b)*;*NEGRdia(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *NEGRdia-PR-imp-NEGRdia*: [|*NEGRdia(a,b)*;*PR(b,c)*|] ==> *NEGRdia(a,c)*
⟨*proof*⟩

theorem *SpR-and-SpR-and-RSSR-imp-RSSRdia*: [|*SpR(a)*;*SpR(b)*;*RSSR(a,b)*|] ==> *RSSRdia(a,b)*
⟨*proof*⟩

theorem *SpR-and-SpR-and-RSSRdia-imp-RSSR*: [|*SpR(a)*;*SpR(b)*;*RSSRdia(a,b)*|] ==> *RSSR(a,b)*
⟨*proof*⟩

theorem *SpR-and-SpR-imp-RSSR-iff-RSSRdia*: [|*SpR(a)*;*SpR(b)*|] ==> (*RSSR(a,b)* <−> *RSSRdia(a,b)*)
⟨*proof*⟩

end

theory *QDistR*

imports *QSizeR RBG QDiaSizeR*

begin

consts

CLR :: Rg => Rg => o
SCLR :: Rg => Rg => o
NR :: Rg => Rg => o
SNR :: Rg => Rg => o
AR :: Rg => Rg => o
FAR :: Rg => Rg => o
MAR :: Rg => Rg => o

SpShR :: Rg => o

defs

CLR-def: CLR(a,b) == (EX c. (SpR(c) & CGR(c,a) & CGR(c,b) & NEGR(c,a)))
SCLR-def: SCLR(a,b) == ~CGR(a,b) & CLR(a,b)
NR-def: NR(a,b) == (EX c. (SpR(c) & CGR(c,a) & CGR(c,b) & (NEGR(c,a) | RSSR(c,a))))
SNR-def: SNR(a,b) == ~CLR(a,b) & NR(a,b)
AR-def: AR(a,b) == ~NR(a,b)
FAR-def: FAR(a,b) == (ALL c. (SpR(c) & CGR(c,a) & CGR(c,b) --> (NEGR(a,c))))
MAR-def: MAR(a,b) == AR(a,b) & ~ FAR(a,b)

SpShR-def: SpShR(a) == (EX b. (MinBSpR(b,a) & RSSR(a,b)))

axioms

PR-imp-NR-imp-NR: PR(a,b) ==> (ALL c. (NR(a,c) --> NR(b,c)))
LRSSR-and-NR-imp-NR: [|LRSSR(a,b);NR(a,b)|] ==> NR(b,a)

lemma SSR-or-notSSR: ALL a b. (SSR(a,b) | ~SSR(a,b))
⟨proof⟩

theorem CGR-imp-CLR: CGR(a,b) ==> CLR(a,b)
⟨proof⟩

theorem SCLR-imp-CLR: SCLR(a,b) ==> CLR(a,b)
⟨proof⟩

theorem *CLR-imp-CGR-or-SCLR*: *CLR(a,b) ==> (CGR(a,b) | SCLR(a,b))*
⟨*proof*⟩

theorem *CGR-imp-notSCLR*: *CGR(a,b) ==> ~SCLR(a,b)*
⟨*proof*⟩

theorem *CLR-imp-NR*: *CLR(a,b) ==> NR(a,b)*
⟨*proof*⟩

theorem *SNR-imp-NR*: *SNR(a,b) ==> NR(a,b)*
⟨*proof*⟩

theorem *NR-imp-CLR-or-SNR*: *NR(a,b) ==> (CLR(a,b) | SNR(a,b))*
⟨*proof*⟩

theorem *CLR-imp-notSNR*: *CLR(a,b) ==> ~SNR(a,b)*
⟨*proof*⟩

theorem *FAR-imp-AR*: *FAR(a,b) ==> AR(a,b)*
⟨*proof*⟩

theorem *MAR-imp-AR*: *MAR(a,b) ==> AR(a,b)*
⟨*proof*⟩

theorem *AR-imp-MAR-or-FAR*: *AR(a,b) ==> (MAR(a,b) | FAR(a,b))*
⟨*proof*⟩

theorem *MAR-imp-notFAR*: *MAR(a,b) ==> ~FAR(a,b)*
⟨*proof*⟩

theorem *NR-or-AR*: *NR(a,b) | AR(a,b)*
⟨*proof*⟩

theorem *CLR-refl*: *(ALL a. CLR(a,a))*
⟨*proof*⟩

theorem *NR-refl*: *(ALL a. NR(a,a))*
⟨*proof*⟩

theorem *SCLR-irrefl*: *(ALL a. ~SCLR(a,a))*
⟨*proof*⟩

theorem *SNR-irrefl*: *(ALL a. ~SNR(a,a))*
⟨*proof*⟩

theorem *AR-irrefl*: (*ALL a.* ~*AR*(*a*,*a*))
⟨*proof*⟩

theorem *FAR-irrefl*: (*ALL a.* ~*FAR*(*a*,*a*))
⟨*proof*⟩

theorem *CLR-and-PR-imp-CLR*: [|*CLR*(*a*,*b*);*PR*(*b*,*c*)|] ==> *CLR*(*a*,*c*)
⟨*proof*⟩

theorem *NR-and-PR-imp-NR*: [|*NR*(*a*,*b*);*PR*(*b*,*c*)|] ==> *NR*(*a*,*c*)
⟨*proof*⟩

theorem *PR-and-notNR-imp-notNR*: [|*PR*(*a*,*b*);~*NR*(*b*,*c*)|] ==> ~*NR*(*a*,*c*)
⟨*proof*⟩

theorem *PR-imp-NR*: *PR*(*a*,*b*) ==> (*NR*(*a*,*b*) & *NR*(*b*,*a*))
⟨*proof*⟩

theorem *PR-and-AR-imp-AR*: [|*PR*(*a*,*b*);*AR*(*b*,*c*)|] ==> *AR*(*a*,*c*)
⟨*proof*⟩

theorem *LRSSR-and-CLR-imp-CLR*: [|*LRSSR*(*a*,*b*);*CLR*(*a*,*b*)|] ==> *CLR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-SCLR-imp-SCLR*: [|*LRSSR*(*a*,*b*);*SCLR*(*a*,*b*)|] ==> *SCLR*(*b*,*a*)
⟨*proof*⟩

theorem *RSSR-and-SNR-imp-SNR*: [|*RSSR*(*a*,*b*);*SNR*(*a*,*b*)|] ==> *SNR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-SNR-imp-NR*: [|*LRSSR*(*a*,*b*);*SNR*(*a*,*b*)|] ==> *NR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-AR-imp-AR*: [|*LRSSR*(*b*,*a*);*AR*(*a*,*b*)|] ==> *AR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-FAR-imp-FAR*: [|*LRSSR*(*b*,*a*);*FAR*(*a*,*b*)|] ==> *FAR*(*b*,*a*)
⟨*proof*⟩

theorem *LRSSR-and-MAR-imp-AR*: [|*LRSSR*(*b*,*a*);*MAR*(*a*,*b*)|] ==> *AR*(*b*,*a*)
⟨*proof*⟩

theorem *RSSR-and-MAR-imp-MAR*: [|*RSSR*(*a*,*b*);*MAR*(*a*,*b*)|] ==> *MAR*(*b*,*a*)
⟨*proof*⟩

end

theory *TNEMO* **imports** *FOL*

begin

typedecl *Ob*
typedecl *Ti*

arities *Ob* :: *term*
Ti :: *term*

consts

O :: *Ob* => *Ob* => *Ti* => *o*
P :: *Ob* => *Ob* => *Ti* => *o*
PP :: *Ob* => *Ob* => *Ti* => *o*
E :: *Ob* => *Ti* => *o*
Me :: *Ob* => *Ob* => *Ti* => *o*

pP :: *Ob* => *Ob* => *o*
pPP :: *Ob* => *Ob* => *o*
pO :: *Ob* => *Ob* => *o*
pMe :: *Ob* => *Ob* => *o*

cP :: *Ob* => *Ob* => *o*
bP :: *Ob* => *Ob* => *o*

axioms

P-exists1: (*ALL x*. (*EX t*. *E*(*x*,*t*)))
P-exists2: (*ALL x y t*. (*P*(*x*,*y*,*t*) −−>(*E*(*x*,*t*) & *E*(*y*,*t*))))
P-trans: (*ALL x y z t*. (*P*(*x*,*y*,*t*) & *P*(*y*,*z*,*t*) −−> *P*(*x*,*z*,*t*)))

P-ssuppl: (ALL x y t. ((E(x,t) & ~P(x,y,t)) --> (EX z. (P(z,x,t) & ~O(z,y,t)))))

defs

E-def: E(x,t) == P(x,x,t)
O-def: O(x,y,t) == (EX z. (P(z,x,t) & P(z,y,t)))
PP-def: PP(x,y,t) == P(x,y,t) & ~P(y,x,t)
Me-def: Me(x,y,t) == (P(x,y,t) & P(y,x,t))

pP-def: pP(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> P(x,y,t)))
pPP-def: pPP(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> PP(x,y,t)))
pO-def: pO(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> O(x,y,t)))
pMe-def: pMe(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> Me(x,y,t)))

cP-def: cP(x,y) == (ALL t. (E(y,t) --> P(x,y,t)))
bP-def: bP(x,y) == (ALL t. (E(x,t) --> P(x,y,t)))

lemma *P-exists2-rule*: P(x,y,t) ==>(E(x,t) & E(y,t))
⟨*proof*⟩

lemma *P-trans-rule*: [|P(x,y,t); P(y,z,t)|] ==> P(x,z,t)
⟨*proof*⟩

lemma *P-Me-rule*: [|P(x,y,t);P(y,x,t)|] ==> Me(x,y,t)
⟨*proof*⟩

lemma *P-ssuppl-rule*: [|E(x,t);~P(x,y,t)|] ==> (EX z. (P(z,x,t) & ~O(z,y,t)))
⟨*proof*⟩

lemma *ltb2*: (~(A & ~B) ==> (~A | B))
⟨*proof*⟩

lemma *P-ssuppl-rule-transpos*: ~(EX z. (P(z,x,t) & ~O(z,y,t))) ==> (~E(x,t) | P(x,y,t))
⟨*proof*⟩

theorem *P-refl*: E(x,t) ==> P(x,x,t)
⟨*proof*⟩

theorem *Me-refl*: E(x,t) ==> Me(x,x,t)
⟨*proof*⟩

theorem *Me-exists2*: Me(x,y,t) ==> (E(x,t) & E(y,t))
⟨*proof*⟩

theorem *Me-sym*: Me(x,y,t) ==> Me(y,x,t)
⟨*proof*⟩

theorem *Me-trans*: [|Me(x,y,t); Me(y,z,t)|] ==> Me(x,z,t)
⟨*proof*⟩

theorem *O-refl*: (ALL x t. E(x,t) --> O(x,x,t))
⟨*proof*⟩

lemma *O-refl-rule*: E(x,t) ==> O(x,x,t)
⟨*proof*⟩

theorem *O-sym*: (ALL x y t. (O(x,y,t) --> O(y,x,t)))
⟨*proof*⟩

lemma *O-sym-rule*: O(x,y,t) ==> O(y,x,t)
⟨*proof*⟩

theorem *O-imp-E-and-E*: O(x,y,t) ==> (E(x,t) & E(y,t))
⟨*proof*⟩

theorem *PP-imp-P*: (ALL x y t. (PP(x,y,t) --> P(x,y,t)))
⟨*proof*⟩

lemma *PP-imp-P-rule*: PP(x,y,t) ==> P(x,y,t)
⟨*proof*⟩

theorem *P-imp-Me-or-PP*: P(x,y,t) ==> (Me(x,y,t) | PP(x,y,t))
⟨*proof*⟩

theorem *PP-or-Me-imp-P*: (PP(x,y,t) | Me(x,y,t)) ==> P(x,y,t)
⟨*proof*⟩

theorem *PP-asym*: (ALL x y t. PP(x,y,t) --> ~PP(y,x,t))
⟨*proof*⟩

lemma *PP-asym-rule*: PP(x,y,t) ==> ~PP(y,x,t)
⟨*proof*⟩

theorem *PP-trans*: (*ALL x y z t*. (*PP*(*x*,*y*,*t*) & *PP*(*y*,*z*,*t*)-->*PP*(*x*,*z*,*t*)))
⟨*proof*⟩

lemma *PP-trans-rule*: **assumes** *p1*: *PP*(*x*,*y*,*t*) **assumes** *p2*: *PP*(*y*,*z*,*t*) **shows** *PP*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *P-and-PP-imp-PP*: [|*P*(*x*,*y*,*t*);*PP*(*y*,*z*,*t*)|] ==> *PP*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *PP-and-P-imp-PP*: [|*PP*(*x*,*y*,*t*);*P*(*y*,*z*,*t*)|] ==> *PP*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *P-and-notMe-imp-PP*: [|*P*(*x*,*y*,*t*);~*Me*(*x*,*y*,*t*)|] ==> *PP*(*x*,*y*,*t*)
⟨*proof*⟩

theorem *P-imp-O*: *P*(*x*,*y*,*t*) ==> *O*(*x*,*y*,*t*)
⟨*proof*⟩

theorem *P-and-O*: [|*P*(*x*,*y*,*t*);*O*(*x*,*z*,*t*)|] ==> *O*(*y*,*z*,*t*)
⟨*proof*⟩

theorem *O-imp-O-imp-P*: (*ALL x y t*. (*E*(*x*,*t*) & (*ALL z*. (*O*(*z*,*x*,*t*) --> *O*(*z*,*y*,*t*))) --> *P*(*x*,*y*,*t*)))
⟨*proof*⟩

lemma *O-imp-O-imp-P-rule*: [|*E*(*x*,*t*);(*ALL z*. (*O*(*z*,*x*,*t*) --> *O*(*z*,*y*,*t*)))|] ==> *P*(*x*,*y*,*t*)
⟨*proof*⟩

lemma *ltb1*: (*ALL z*. (*A*(*z*,*x*,*t*) & *B*(*z*,*y*,*t*))) ==> (*ALL z*. *A*(*z*,*x*,*t*)) & (*ALL z*. *B*(*z*,*y*,*t*))
⟨*proof*⟩

theorem *O-iff-O-iff-Me* : (*ALL x y t*. ((*E*(*x*,*t*) & *E*(*y*,*t*) & (*ALL z*. (*O*(*z*,*x*,*t*) <-> *O*(*z*,*y*,*t*)))) <-> *Me*(*x*,*y*,*t*)))
⟨*proof*⟩

theorem *P-iff-P-iff-Me*: (*ALL x y t*. ((*E*(*x*,*t*) & *E*(*y*,*t*) & (*ALL z*. (*P*(*z*,*x*,*t*)

<-> P(z,y,t)))) <-> Me(x,y,t)))
⟨proof⟩

theorem *pP-refl*: *ALL x. pP(x,x)*
⟨proof⟩

theorem *pP-trans*: [|*pP(x,y);pP(y,z)*|] ==> *pP(x,z)*
⟨proof⟩

theorem *pPP-asym*: *pPP(x,y)* ==> ~*pPP(y,x)*
⟨proof⟩

theorem *pPP-trans*: [|*pPP(x,y);pPP(y,z)*|] ==> *pPP(x,z)*
⟨proof⟩

theorem *pPP-imp-pP-and-notpP*: *pPP(x,y)* ==> (*pP(x,y)* & ~*pP(y,x)*)
⟨proof⟩

theorem *pO-refl*: (*ALL x. pO(x,x)*)
⟨proof⟩

theorem *pO-sym*: *pO(x,y)* ==> *pO(y,x)*
⟨proof⟩

theorem *SharedpP-imp-pO*: (*EX z.* (*pP(z,x)* & *pP(z,y)*)) ==> *pO(x,y)*
⟨proof⟩

theorem *cP-refl*: (*ALL x.* (*cP(x,x)*))
⟨proof⟩

theorem *cP-trans*: [|*cP(x,y);cP(y,z)*|] ==> *cP(x,z)*
⟨proof⟩

theorem *bP-refl*: (*ALL x.* (*bP(x,x)*))
⟨proof⟩

theorem *bP-trans*: [|*bP(x,y);bP(y,z)*|] ==> *bP(x,z)*

⟨*proof*⟩

theorem *pP-imp-cP-and-bP*: *pP(x,y) ==> (cP(x,y) & bP(x,y))*
⟨*proof*⟩

theorem *cP-and-bP-imp-pP*: *[|cP(x,y);bP(x,y)|] ==> pP(x,y)*
⟨*proof*⟩

end
theory *TORL*

imports *TNEMO EMR*

begin

consts

L :: *Ob => Rg => Ti => o*
LocIn :: *Ob => Ob => Ti => o*
PCoin :: *Ob => Ob => Ti => o*
ContIn :: *Ob => Ob => Ti => o*

pLocIn :: *Ob => Ob => o*
pPCoin :: *Ob => Ob => o*
pContIn :: *Ob => Ob => o*

axioms

L-exists: *(ALL x t. (E(x,t) <−> (EX a. L(x,a,t))))*
L-P-PR: *(ALL x y a b t. (L(x,a,t) & L(y,b,t) & P(x,y,t) −−> PR(a,b)))*
P-and-L-and-L-imp-P: *(ALL x y a t. (P(x,y,t) & L(x,a,t) & L(y,a,t) −−> P(y,x,t)))*

defs

LocIn-def: *LocIn(x,y,t) == (EX a b. (L(x,a,t) & L(y,b,t) & PR(a,b)))*
PCoin-def: *PCoin(x,y,t) == (EX a b. (L(x,a,t) & L(y,b,t) & OR(a,b)))*
ContIn-def: *ContIn(x,y,t) == LocIn(x,y,t) & ~O(x,y,t)*

pLocIn-def: *pLocIn(x,y) == (ALL t. ((E(x,t) | E(y,t)) −−> LocIn(x,y,t)))*
pPCoin-def: *pPCoin(x,y) == (ALL t. ((E(x,t) | E(y,t)) −−> PCoin(x,y,t)))*
pContIn-def: *pContIn(x,y) == (ALL t. ((E(x,t) | E(y,t)) −−> ContIn(x,y,t)))*

lemma *L-exists1*: *E(x,t) ==> (EX a. L(x,a,t))*
⟨*proof*⟩

lemma *L-exists2*: *(EX a. L(x,a,t)) ==> E(x,t)*
⟨*proof*⟩

lemma *L-P-PR-rule*: *[|L(x,a,t); L(y,b,t); P(x,y,t)|] ==> PR(a,b)*
⟨*proof*⟩

lemma *P-and-L-and-L-imp-P-rule*: *[|P(x,y,t);L(x,a,t);L(y,a,t)|] ==> P(y,x,t)*
⟨*proof*⟩

theorem *L-unique*: *[|L(x,a,t);L(x,b,t)|] ==> a=b*
⟨*proof*⟩

theorem *LocIn-imp-E-and-E*: *LocIn(x,y,t) ==> (E(x,t) & E(y,t))*
⟨*proof*⟩

theorem *P-imp-L*: *P(x,y,t) ==> (EX a b. (L(x,a,t) & L(y,b,t) & PR(a,b)))*
⟨*proof*⟩

theorem *L-and-L-and-PR-imp-L*: *[|L(x,a,t);L(y,b,t);PR(a,b)|] ==> LocIn(x,y,t)*
⟨*proof*⟩

theorem *LocIn-imp-PCoin*: *LocIn(x,y,t) ==> PCoin(x,y,t)*
⟨*proof*⟩

theorem *E-imp-LocIn*: *E(x,t) ==> LocIn(x,x,t)*
⟨*proof*⟩

theorem *LocIn-trans*: *[|LocIn(x,y,t);LocIn(y,z,t)|] ==> LocIn(x,z,t)*
⟨*proof*⟩

theorem *PCoin-sym*: *PCoin(x,y,t) ==> PCoin(y,x,t)*
⟨*proof*⟩

theorem *P-imp-LocIn*: *P(x,y,t) ==> LocIn(x,y,t)*
⟨*proof*⟩

theorem *P-and-LocIn-imp-P*: *[|P(x,y,t);LocIn(y,x,t)|] ==> P(y,x,t)*
⟨*proof*⟩

theorem *LocIn-and-P-imp-LocIn*: *[|LocIn(x,y,t);P(y,z,t)|] ==> LocIn(x,z,t)*

⟨*proof*⟩

theorem *P-and-LocIn-imp-LocIn*: [|$P(x,y,t)$;$LocIn(y,z,t)$|] ==> $LocIn(x,z,t)$
⟨*proof*⟩

theorem *O-imp-PCoin*: $O(x,y,t)$ ==> $PCoin(x,y,t)$
⟨*proof*⟩

theorem *pLocIn-imp-pPCoin*: $pLocIn(x,y)$ ==> $pPCoin(x,y)$
⟨*proof*⟩

theorem *pLocIn-refl*: (*ALL x*. $pLocIn(x,x)$)
⟨*proof*⟩

theorem *pLocIn-trans*: [|$pLocIn(x,y)$;$pLocIn(y,z)$|] ==> $pLocIn(x,z)$
⟨*proof*⟩

theorem *pPCoin-sym*: $pPCoin(x,y)$ ==> $pPCoin(y,x)$
⟨*proof*⟩

theorem *pP-imp-pLocIn*: $pP(x,y)$ ==> $pLocIn(x,y)$
⟨*proof*⟩

theorem *pLocIn-and-pP-imp-pLocIn*: [|$pLocIn(x,y)$;$pP(y,z)$|] ==> $pLocIn(x,z)$
⟨*proof*⟩

theorem *pP-and-pLocIn-imp-pLocIn*: [|$pP(x,y)$;$pLocIn(y,z)$|] ==> $pLocIn(x,z)$
⟨*proof*⟩

theorem *pP-and-pLocIn-imp-pP*: [|$pP(x,y)$;$pLocIn(y,x)$|] ==> $pP(y,x)$
⟨*proof*⟩

theorem *pO-imp-pPCoin*: $pO(x,y)$ ==> $pPCoin(x,y)$
⟨*proof*⟩

theorem *pContIn-imp-pLocIn-and-notpO*: $pContIn(x,y)$ ==> $pLocIn(x,y)$ & ~$pO(x,y)$
⟨*proof*⟩

theorem *pContIn-imp-pLocIn-and-notO*: $pContIn(x,y)$ ==> ($pLocIn(x,y)$ & (*ALL t*. ~$O(x,y,t)$))
⟨*proof*⟩

theorem *pLocIn-and-notO-imp-pContIn*: [|*pLocIn*(*x*,*y*);(*ALL t*. ~*O*(*x*,*y*,*t*))|] ==>
pContIn(*x*,*y*)
⟨*proof*⟩

end

theory *QSizeO*

imports *TORL QSizeR*

begin

consts

SS :: *Ob* => *Ob* => *Ti* => *o*
LE :: *Ob* => *Ob* => *Ti* => *o*
RSS :: *Ob* => *Ob* => *Ti* => *o*
NEG :: *Ob* => *Ob* => *Ti* => *o*
SSC :: *Ob* => *Ob* => *Ti* => *o*

pSS :: *Ob* => *Ob* => *o*
pLE :: *Ob* => *Ob* => *o*
pRSS :: *Ob* => *Ob* => *o*
pNEG :: *Ob* => *Ob* => *o*
pSSC :: *Ob* => *Ob* => *o*

defs

SS-def: *SS*(*x*,*y*,*t*) == (*EX a b*. (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *SSR*(*a*,*b*)))
LE-def: *LE*(*x*,*y*,*t*) == (*EX a b*. (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *LER*(*a*,*b*)))
RSS-def: *RSS*(*x*,*y*,*t*) == (*EX a b*. (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *RSSR*(*a*,*b*)))
NEG-def: *NEG*(*x*,*y*,*t*) == (*EX a b*. (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *NEGR*(*a*,*b*)))
SSC-def: *SSC*(*x*,*y*,*t*) == (*EX a b*. (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *SSCR*(*a*,*b*)))

pSS-def: *pSS*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) --> *SS*(*x*,*y*,*t*)))
pLE-def: *pLE*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) --> *LE*(*x*,*y*,*t*)))
pRSS-def: *pRSS*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) --> *RSS*(*x*,*y*,*t*)))
pNEG-def: *pNEG*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) --> *NEG*(*x*,*y*,*t*)))
pSSC-def: *pSSC*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) --> *SSC*(*x*,*y*,*t*)))

theorem *SS-imp-E-and-E*: *SS*(*x*,*y*,*t*) ==> (*E*(*x*,*t*) & *E*(*y*,*t*))
⟨*proof*⟩

theorem *SS-refl*: *ALL x t*. (*E*(*x*,*t*) <-> *SS*(*x*,*x*,*t*))

⟨proof⟩

theorem *SS-sym*: *SS(x,y,t) ==> SS(y,x,t)*
⟨proof⟩

theorem *SS-trans*: [|*SS(x,y,t);SS(y,z,t)*|] ==> *SS(x,z,t)*
⟨proof⟩

theorem *P-and-SS-imp-P*: [|*P(x,y,t);SS(x,y,t)*|] ==> *P(y,x,t)*
⟨proof⟩

theorem *P-and-P-imp-SS*: [|*P(x,y,t);P(y,x,t)*|] ==> *SS(x,y,t)*
⟨proof⟩

theorem *LE-total*: [|*E(x,t);E(y,t)*|] ==> (*LE(x,y,t)* | *LE(y,x,t)*)
⟨proof⟩

theorem *LE-and-LE-imp-SS*: [|*LE(x,y,t);LE(y,x,t)*|] ==> *SS(x,y,t)*
⟨proof⟩

theorem *LE-imp-E-and-E*: *LE(x,y,t)* ==> (*E(x,t)* & *E(y,t)*)
⟨proof⟩

theorem *LE-refl*: *ALL x t. E(x,t)* <-> *LE(x,x,t)*
⟨proof⟩

theorem *LE-trans*: [|*LE(x,y,t);LE(y,z,t)*|] ==> *LE(x,z,t)*
⟨proof⟩

theorem *LE-and-LE-imp-SS*: [|*LE(x,y,t);LE(y,x,t)*|] ==> *SS(x,y,t)*
⟨proof⟩

theorem *SS-and-LE-imp-LE*: [|*SS(x,y,t);LE(y,z,t)*|] ==> *LE(x,z,t)*
⟨proof⟩

theorem *LE-and-SS-imp-LE*: [|*LE(x,y,t);SS(y,z,t)*|] ==> *LE(x,z,t)*
⟨proof⟩
thm *ssubst*
⟨proof⟩

theorem *RSS-imp-E-and-E*: *RSS(x,y,t)* ==> (*E(x,t)* & *E(y,t)*)
⟨proof⟩

theorem *RSS-refl*: *ALL x t.* (*E*(*x*,*t*) <−> *RSS*(*x*,*x*,*t*))
⟨*proof*⟩

theorem *RSS-sym*: *RSS*(*x*,*y*,*t*) ==> *RSS*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *RSS-and-SS-imp-RSS*: [|*RSS*(*x*,*y*,*t*);*SS*(*y*,*z*,*t*)|] ==> *RSS*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *SS-and-RSS-imp-RSS*: [|*SS*(*x*,*y*,*t*);*RSS*(*y*,*z*,*t*)|] ==> *RSS*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *NEG-imp-LE*: *NEG*(*x*,*y*,*t*) ==> *LE*(*x*,*y*,*t*)
⟨*proof*⟩

theorem *NEG-imp-E-and-E*: *NEG*(*x*,*y*,*t*) ==>(*E*(*x*,*t*) & *E*(*y*,*t*))
⟨*proof*⟩

theorem *NEG-irrefl*: *ALL x t.* ~*NEG*(*x*,*x*,*t*)
⟨*proof*⟩

theorem *NEG-assym*: *NEG*(*x*,*y*,*t*) ==> ~*NEG*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *NEG-trans*: [|*NEG*(*x*,*y*,*t*);*NEG*(*y*,*z*,*t*)|] ==> *NEG*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *NEG-and-LE-imp-NEG*: [|*NEG*(*x*,*y*,*t*);*LE*(*y*,*z*,*t*)|] ==> *NEG*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *LE-and-NEG-imp-NEG*: [|*LE*(*x*,*y*,*t*);*NEG*(*y*,*z*,*t*)|] ==> *NEG*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *P-imp-LE*: *P*(*x*,*y*,*t*) ==> *LE*(*x*,*y*,*t*)
⟨*proof*⟩

theorem *NEG-and-P-imp-NEG*: [|*NEG*(*x*,*y*,*t*);*P*(*y*,*z*,*t*)|] ==> *NEG*(*x*,*z*,*t*)
⟨*proof*⟩

theorem *P-and-NEG-imp-NEG*: [|*P*(*x*,*y*,*t*);*NEG*(*y*,*z*,*t*)|] ==> *NEG*(*x*,*z*,*t*)
⟨*proof*⟩
thm *LE-and-NEG-imp-NEG*
⟨*proof*⟩

theorem *SSC-refl*: *ALL x t.* (*E*(*x*,*t*) <−> *SSC*(*x*,*x*,*t*))

⟨*proof*⟩

theorem *SSC-sym*: *SSC*(*x*,*y*,*t*) ==> *SSC*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *pSS-refl*: *ALL x*. *pSS*(*x*,*x*)
⟨*proof*⟩

theorem *pSS-sym*: *pSS*(*x*,*y*) ==> *pSS*(*y*,*x*)
⟨*proof*⟩

theorem *pSS-trans*: [|*pSS*(*x*,*y*);*pSS*(*y*,*z*)|] ==> *pSS*(*x*,*z*)
⟨*proof*⟩

theorem *pP-and-pSS-imp-pP*: [|*pP*(*x*,*y*);*pSS*(*x*,*y*)|] ==> *pP*(*y*,*x*)
⟨*proof*⟩

theorem *pLE-refl*: *ALL x*. *pLE*(*x*,*x*)
⟨*proof*⟩

theorem *pLE-trans*: [|*pLE*(*x*,*y*);*pLE*(*y*,*z*)|] ==> *pLE*(*x*,*z*)
⟨*proof*⟩

theorem *pLE-and-pLE-imp-pSS*: [|*pLE*(*x*,*y*);*pLE*(*y*,*x*)|] ==> *pSS*(*x*,*y*)
⟨*proof*⟩

theorem *pSS-and-pLE-imp-pLE*: [|*pSS*(*x*,*y*);*pLE*(*y*,*z*)|] ==> *pLE*(*x*,*z*)
⟨*proof*⟩

theorem *pLE-and-pSS-imp-pLE*: [|*pLE*(*x*,*y*);*pSS*(*y*,*z*)|] ==> *pLE*(*x*,*z*)
⟨*proof*⟩

theorem *pRSS-refl*: *ALL x*. *pRSS*(*x*,*x*)
⟨*proof*⟩

theorem *pRSS-sym*: *pRSS*(*x*,*y*) ==> *pRSS*(*y*,*x*)
⟨*proof*⟩

theorem *pRSS-and-pSS-imp-pRSS*: [|*pRSS*(*x*,*y*);*pSS*(*y*,*z*)|] ==> *pRSS*(*x*,*z*)
⟨*proof*⟩

theorem *pSS-and-pRSS-imp-pRSS*: [|*pSS*(*x*,*y*);*pRSS*(*y*,*z*)|] ==> *pRSS*(*x*,*z*)
⟨*proof*⟩

theorem *pNEG-irrefl*: *ALL x*. ~*pNEG*(*x*,*x*)
⟨*proof*⟩

theorem *pNEG-assym*: *pNEG*(*x*,*y*) ==> ~*pNEG*(*y*,*x*)
⟨*proof*⟩

theorem *pNEG-trans*: [|*pNEG*(*x*,*y*);*pNEG*(*y*,*z*)|] ==> *pNEG*(*x*,*z*)
⟨*proof*⟩

theorem *pNEG-and-pLE-imp-pNEG*: [|*pNEG*(*x*,*y*);*pLE*(*y*,*z*)|] ==> *pNEG*(*x*,*z*)
⟨*proof*⟩

theorem *pLE-and-pNEG-imp-pNEG*: [|*pLE*(*x*,*y*);*pNEG*(*y*,*z*)|] ==> *pNEG*(*x*,*z*)
⟨*proof*⟩

theorem *pP-imp-pLE*: *pP*(*x*,*y*) ==> *pLE*(*x*,*y*)
⟨*proof*⟩

theorem *pNEG-and-pP-imp-pNEG*: [|*pNEG*(*x*,*y*);*pP*(*y*,*z*)|] ==> *pNEG*(*x*,*z*)
⟨*proof*⟩

theorem *pP-and-pNEG-imp-pNEG*: [|*pP*(*x*,*y*);*pNEG*(*y*,*z*)|] ==> *pNEG*(*x*,*z*)
⟨*proof*⟩

theorem *pSSC-refl*: *ALL x*. *pSSC*(*x*,*x*)
⟨*proof*⟩

theorem *pSSC-sym*: *pSSC*(*x*,*y*) ==> *pSSC*(*y*,*x*)
⟨*proof*⟩

end
theory *TMTL*

imports *TNEMO TORL RBG*

begin

consts

C :: Ob => Ob => Ti => o
EC :: Ob => Ob => Ti => o
DC :: Ob => Ob => Ti => o
pC :: Ob => Ob => o
pEC :: Ob => Ob => o
pDC :: Ob => Ob => o

defs

C-def: C(x,y,t) == (EX a b. (L(x,a,t) & L(y,b,t) & CGR(a,b)))
EC-def: EC(x,y,t) == (EX a b. (L(x,a,t) & L(y,b,t) & ECR(a,b)))
DC-def: DC(x,y,t) == (EX a b. (L(x,a,t) & L(y,b,t) & DCR(a,b)))
pC-def: pC(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> C(x,y,t)))
pEC-def: pEC(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> EC(x,y,t)))
pDC-def: pDC(x,y) == (ALL t. ((E(x,t) | E(y,t)) --> DC(x,y,t)))

theorem E-imp-C: E(x,t) ==> C(x,x,t)
⟨proof⟩

theorem C-sym: C(x,y,t) ==> C(y,x,t)
⟨proof⟩

theorem C-imp-E-and-E: C(x,y,t) ==> (E(x,t) & E(y,t))
⟨proof⟩

theorem P-imp-C-imp-C: P(x,y,t) ==> (ALL z. (C(z,x,t) --> C(z,y,t)))
⟨proof⟩

theorem L-and-L-and-C-imp-CGR: [|L(x,a,t);L(y,b,t);CGR(a,b)|] ==> C(x,y,t)
⟨proof⟩

theorem EC-imp-C-and-notO: EC(x,y,t) ==> (C(x,y,t) & ~O(x,y,t))
⟨proof⟩

theorem EC-irrefl: (ALL x t. (~EC(x,x,t)))
⟨proof⟩

theorem EC-sym: EC(x,y,t) ==> EC(y,x,t)
⟨proof⟩

theorem *DC-irrefl*: *(ALL x t. (~DC(x,x,t)))*
⟨*proof*⟩

theorem *DC-sym*: *DC(x,y,t) ==> DC(y,x,t)*
⟨*proof*⟩

theorem *DC-imp-notC*: *DC(x,y,t) ==> ~C(x,y,t)*
⟨*proof*⟩

theorem *P-imp-C*: *P(x,y,t) ==> C(x,y,t)*
⟨*proof*⟩

theorem *O-imp-C*: *O(x,y,t) ==> C(x,y,t)*
⟨*proof*⟩

theorem *P-and-C-imp-C*: *[|P(x,y,t);C(x,z,t)|] ==> C(y,z,t)*
⟨*proof*⟩

theorem *PCoin-imp-C*: *PCoin(x,y,t) ==> C(x,y,t)*
⟨*proof*⟩

theorem *LocIn-imp-C*: *LocIn(x,y,t) ==> C(x,y,t)*
⟨*proof*⟩

theorem *PCoin-or-EC-or-notC*: *(ALL x y t. (PCoin(x,y,t) | EC(x,y,t) | ~C(x,y,t)))*
⟨*proof*⟩

theorem *C-and-LocIn-imp-C*: *[|C(x,y,t);LocIn(y,z,t)|] ==> C(x,z,t)*
⟨*proof*⟩

theorem *pEC-irrefl*: *ALL x. ~pEC(x,x)*
⟨*proof*⟩

theorem *pEC-sym*: *pEC(x,y) ==> pEC(y,x)*
⟨*proof*⟩

theorem *pDC-irrefl*: *ALL x. ~pDC(x,x)*
⟨*proof*⟩

theorem *pDC-sym*: *pDC(x,y) ==> pDC(y,x)*
⟨*proof*⟩

theorem *pDC-imp-notC*: *pDC(x,y) ==> (ALL t. ~ C(x,y,t))*

⟨*proof*⟩

theorem *pDC-imp-not-pC*: *pDC*(*x*,*y*) ==> ~ *pC*(*x*,*y*)
⟨*proof*⟩

theorem *pP-imp-pC*: *pP*(*x*,*y*) ==> *pC*(*x*,*y*)
⟨*proof*⟩

theorem *pPCoin-imp-pC*: *pPCoin*(*x*,*y*) ==> *pC*(*x*,*y*)
⟨*proof*⟩

theorem *pC-and-pLocIn-imp-pC*: [|*pC*(*x*,*y*);*pLocIn*(*y*,*z*)|] ==> *pC*(*x*,*z*)
⟨*proof*⟩

theorem *pEC-imp-pC-and-notpO*: *pEC*(*x*,*y*) ==> (*pC*(*x*,*y*) & ~*pO*(*x*,*y*))
⟨*proof*⟩

theorem *pP-imp-pC-imp-pC*: *pP*(*x*,*y*) ==> (*ALL z*. (*pC*(*z*,*x*) --> *pC*(*z*,*y*)))
⟨*proof*⟩

theorem *pC-sym*: *pC*(*x*,*y*) ==> *pC*(*y*,*x*)
⟨*proof*⟩

theorem *pO-imp-pC*: *pO*(*x*,*y*) ==> *pC*(*x*,*y*)
⟨*proof*⟩

theorem *pP-and-pC-imp-pC*: [|*pP*(*x*,*y*);*pC*(*x*,*z*)|] ==> *pC*(*y*,*z*)
⟨*proof*⟩

end

theory *Adjacency*

imports *QSizeR QSizeO RBG TORL TMTL*

begin

consts

rAdj :: *Rg* => *Rg* => *o*

Adj :: *Ob* => *Ob* => *Ti* => *o*
pAdj :: *Ob* => *Ob* => *o*
Att :: *Ob* => *Ob* => *o*

defs

rAdj-def: *rAdj*(*a*,*b*) == ~*CGR*(*a*,*b*) & (*EX c*. (*SpR*(*c*) & *NEGR*(*c*,*a*) & *NEGR*(*c*,*b*) & *CGR*(*c*,*a*) & *CGR*(*c*,*b*)))

Adj-def: *Adj*(*x*,*y*,*t*) == (*EX a b* . (*L*(*x*,*a*,*t*) & *L*(*y*,*b*,*t*) & *rAdj*(*a*,*b*)))
pAdj-def: *pAdj*(*x*,*y*) == (*ALL t*. ((*E*(*x*,*t*) | *E*(*y*,*t*)) −−> *Adj*(*x*,*y*,*t*)))

Att-def: *Att*(*x*,*y*) == *pDC*(*x*,*y*) & (*EX z1 z2*. (*pPP*(*z1*,*x*) & *pPP*(*z2*,*y*) & *pNEG*(*z1*,*x*) & *pNEG*(*z2*,*y*) & *pAdj*(*z1*,*z2*)))

theorem *rAdj-irrefl*: *ALL a*. (~*rAdj*(*a*,*a*))
⟨*proof*⟩

theorem *rAdj-sym*: *rAdj*(*a*,*b*) ==> *rAdj*(*b*,*a*)
⟨*proof*⟩

theorem *rAdj-and-PR-and-PR-and-notCGR-imp-rAdj*: [|*rAdj*(*a*,*b*);*PR*(*a*,*aa*);*PR*(*b*,*bb*);~*CGR*(*aa*,*bb*)|] ==> *rAdj*(*aa*,*bb*)
⟨*proof*⟩

theorem *Adj-irrefl*: *ALL x t*. (~*Adj*(*x*,*x*,*t*))
⟨*proof*⟩

theorem *Adj-sym*: *Adj*(*x*,*y*,*t*) ==> *Adj*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *Adj-imp-notC*: *Adj*(*x*,*y*,*t*) ==> ~*C*(*x*,*y*,*t*)
⟨*proof*⟩

theorem *Adj-exists*: *Adj*(*x*,*y*,*t*) ==> (*E*(*x*,*t*) & *E*(*y*,*t*))
⟨*proof*⟩

theorem *Adj-and-P-and-P-and-notC-imp-Adj*: [|*Adj*(*x*,*y*,*t*);*P*(*x*,*xx*,*t*);*P*(*y*,*yy*,*t*);~*C*(*xx*,*yy*,*t*)|] ==> *Adj*(*xx*,*yy*,*t*)

⟨*proof*⟩
thm *rAdj-and-PR-and-PR-and-notCGR-imp-rAdj*
⟨*proof*⟩

theorem *pAdj-and-pP-and-pP-and-pDC-imp-pAdj*: [|*pAdj*(*x*,*y*);*pP*(*x*,*xx*);*pP*(*y*,*yy*);*pDC*(*xx*,*yy*)|]
==> *pAdj*(*xx*,*yy*)
⟨*proof*⟩

theorem *pAdj-irrefl*: *ALL x*. (~*pAdj*(*x*,*x*))
⟨*proof*⟩

theorem *pAdj-sym*: *pAdj*(*x*,*y*) ==> *pAdj*(*y*,*x*)
⟨*proof*⟩

theorem *Att-imp-pAdj*: *Att*(*x*,*y*) ==> *pAdj*(*x*,*y*)
⟨*proof*⟩

theorem *Att-sym*: *Att*(*x*,*y*) ==> *Att*(*y*,*x*)
⟨*proof*⟩

theorem *Att-irrefl*: (*ALL x*. ~*Att*(*x*,*x*))
⟨*proof*⟩

end
theory *Collections*

imports *TNEMO*

begin

typedecl *Co*

arities *Co* :: *term*

consts

In :: *Ob* => *Co* => *o*
Union :: *Co* => *Co* => *Co* => *o*
Intersect :: *Co* => *Co* => *Co* => *o*

Subseteq :: *Co* => *Co* => *o*
Fp :: *Co* => *Ti* => *o*
Pp :: *Co* => *Ti* => *o*
Np :: *Co* => *Ti* => *o*

DCo :: *Co* => *Ti* => *o*

axioms

Co-members: (*ALL p.* (*EX x y.* (*In*(*x*,*p*) & *In*(*y*,*p*) & *x* ~= *y*)))
Co-ext: (*ALL p q.* (*p*=*q* <−> (*ALL x.* (*In*(*x*,*p*) <−> *In*(*x*,*q*)))))
Co-union: (*ALL p q.* (*EX r.* (*Union*(*p*,*q*,*r*))))
Co-intersect: (*ALL p q.* (*EX x y.* (*x* ~= *y* & *In*(*x*,*p*) & *In*(*x*,*q*) & *In*(*y*,*p*) & *In*(*y*,*q*))) −−> (*EX r.* (*Intersect*(*p*,*q*,*r*))))

defs

Subseteq-def: *Subseteq*(*p*,*q*) == (*ALL x.* (*In*(*x*,*p*) −−> *In*(*x*,*q*)))
Union-def: *Union*(*p*,*q*,*r*) == (*ALL x.* (*In*(*x*,*r*) <−> (*In*(*x*,*p*) | *In*(*x*,*q*))))
Intersect-def: *Intersect*(*p*,*q*,*r*) == (*ALL x.* (*In*(*x*,*r*) <−> (*In*(*x*,*p*) & *In*(*x*,*q*))))
Fp-def: *Fp*(*p*,*t*) == (*ALL x.* (*In*(*x*,*p*) −−> *E*(*x*,*t*)))
Pp-def: *Pp*(*p*,*t*) == (*EX x.* (*In*(*x*,*p*) & *E*(*x*,*t*)))
Np-def: *Np*(*p*,*t*) == (~*Pp*(*p*,*t*))

DCo-def: *DCo*(*p*,*t*) == (*ALL x y.* (*In*(*x*,*p*) & *In*(*y*,*p*) & *O*(*x*,*y*,*t*) −−> *x*=*y*))

lemma *Co-ext-rule1*: *p*=*q* ==> (*ALL x.* (*In*(*x*,*p*) <−> *In*(*x*,*q*)))
⟨*proof*⟩

lemma *Co-ext-rule2*: (*ALL x.* (*In*(*x*,*p*) <−> *In*(*x*,*q*))) ==> *p*=*q*
⟨*proof*⟩

theorem *Union-unique*: [|*Union*(*p*,*q*,*r1*);*Union*(*p*,*q*,*r2*)|] ==> *r1*=*r2*
⟨*proof*⟩

theorem *Intersect-unique*: [|*Intersect*(*p*,*q*,*r1*);*Intersect*(*p*,*q*,*r2*)|] ==> *r1*=*r2*
⟨*proof*⟩

theorem *Subseteq-refl*: (*ALL p. Subseteq*(*p*,*p*))
⟨*proof*⟩

theorem *Subseteq-antisym*: (*ALL p q*. (*Subseteq*(*p*,*q*) & *Subseteq*(*q*,*p*) −−> *p*=*q*))
⟨*proof*⟩

lemma *Subseteq-antisym-rule*: [|*Subseteq*(*p*,*q*);*Subseteq*(*q*,*p*)|]==>*p*=*q*
⟨*proof*⟩

theorem *Subseteq-trans*: (*ALL p q r*. (*Subseteq*(*p*,*q*) & *Subseteq*(*q*,*r*) −−> *Subseteq*(*p*,*r*)))
⟨*proof*⟩

lemma *Subseteq-trans-rule*: [|*Subseteq*(*p*,*q*); *Subseteq*(*q*,*r*)|] ==> *Subseteq*(*p*,*r*)
⟨*proof*⟩

theorem *Fp-imp-Pp*: *Fp*(*p*,*t*) ==> *Pp*(*p*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-Fp-imp-Fp*: [|*Subseteq*(*p*,*q*);*Fp*(*q*,*t*)|] ==> *Fp*(*p*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-Np-imp-Np*: [|*Subseteq*(*p*,*q*);*Np*(*q*,*t*)|] ==> *Np*(*p*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-Pp-imp-Pp*: [|*Subseteq*(*p*,*q*);*Pp*(*p*,*t*)|] ==> *Pp*(*q*,*t*)
⟨*proof*⟩

theorem *Np-imp-Do*: *Np*(*p*,*t*) ==> *DCo*(*p*,*t*)
⟨*proof*⟩

theorem *DCo-subseteq*: (*ALL p q t*. (*DCo*(*p*,*t*) & *Subseteq*(*q*,*p*) −−> *DCo*(*q*,*t*)))
⟨*proof*⟩

lemma *DCo-subseteq-rule*: [|*DCo*(*p*,*t*); *Subseteq*(*q*,*p*)|] ==> *DCo*(*q*,*t*)
⟨*proof*⟩

end
theory *SumsPartitions*

imports *TNEMO Collections*

begin

consts

SumPp :: *Co* => *Ob* => *Ti* => *o*
PtPp :: *Co* => *Ob* => *Ti* => *o*
SumFp :: *Co* => *Ob* => *Ti* => *o*
PtFp :: *Co* => *Ob* => *Ti* => *o*

cSumFp :: *Co* => *Ob* => *o*
bSumFp :: *Co* => *Ob* => *o*
pSumFp :: *Co* => *Ob* => *o*

defs

SumPp-def: *SumPp*(*p*,*x*,*t*) == (*Pp*(*p*,*t*) & (*ALL z*. (*O*(*z*,*x*,*t*) <−> (*EX y*. (*In*(*y*,*p*) & *O*(*z*,*y*,*t*))))))
SumFp-def: *SumFp*(*p*,*x*,*t*) == (*Fp*(*p*,*t*) & (*ALL z*. (*O*(*z*,*x*,*t*) <−> (*EX y*. (*In*(*y*,*p*) & *O*(*z*,*y*,*t*))))))
PtPp-def: *PtPp*(*p*,*x*,*t*) == (*SumPp*(*p*,*x*,*t*) & *DCo*(*p*,*t*))
PtFp-def: *PtFp*(*p*,*x*,*t*) == (*SumFp*(*p*,*x*,*t*) & *DCo*(*p*,*t*))

cSumFp-def: *cSumFp*(*p*,*x*) == (*ALL t*. (*E*(*x*,*t*) −−> *SumFp*(*p*,*x*,*t*)))
bSumFp-def: *bSumFp*(*p*,*x*) == (*ALL t*. (*Fp*(*p*,*t*) −−> *SumFp*(*p*,*x*,*t*)))
pSumFp-def: *pSumFp*(*p*,*x*) == *cSumFp*(*p*,*x*) & *bSumFp*(*p*,*x*)

theorem *SumPp-and-In-and-E-imp-P*: (*ALL p x y t*. (*SumPp*(*p*,*x*,*t*) & *In*(*y*,*p*) & *E*(*y*,*t*) −−> *P*(*y*,*x*,*t*)))
⟨*proof*⟩

lemma *SumPp-and-In-and-E-imp-P-rule*: [|*SumPp*(*p*,*x*,*t*); *In*(*y*,*p*);*E*(*y*,*t*)|] ==> *P*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *SumFp-and-In-imp-P*: [|*SumFp*(*p*,*x*,*t*); *In*(*y*,*p*)|] ==> *P*(*y*,*x*,*t*)
⟨*proof*⟩

theorem *SumPp-imp-E*: (*ALL p x t*. (*SumPp*(*p*,*x*,*t*) −−> *E*(*x*,*t*)))
⟨*proof*⟩

lemma *SumPp-imp-E-rule*: *SumPp*(*p*,*x*,*t*) ==> *E*(*x*,*t*)
⟨*proof*⟩

theorem *SumPp-and-SumPp-imp-Me*: (*ALL p x y t*. (*SumPp*(*p*,*x*,*t*) & *SumPp*(*p*,*y*,*t*)

--> Me(x,y,t)))
⟨proof⟩

lemma *SumPp-and-SumPp-imp-Me-rule*: [|SumPp(p,x,t);SumPp(p,y,t)|] ==> Me(x,y,t)
⟨proof⟩

theorem *SumPp-and-P-impl-Overlap*: [|SumPp(p,y,t);P(x,y,t)|] ==> (EX z. (In(z,p) & O(x,z,t)))
⟨proof⟩

theorem *SumPp-and-SumPp-and-Subseteq-impl-P*: [|SumPp(p,x,t);SumPp(q,y,t);Subseteq(p,q)|] ==> P(x,y,t)
⟨proof⟩

theorem *SumFp-imp-SumPp*: SumFp(p,x,t) ==> SumPp(p,x,t)
⟨proof⟩

theorem *PtFp-imp-PtPp*: PtFp(p,x,t) ==> PtPp(p,x,t)
⟨proof⟩

theorem *cSumFp-imp-E-imp-Fp*: cSumFp(p,x) ==> (ALL t. (E(x,t) --> Fp(p,t)))
⟨proof⟩

theorem *cSumFp-and-In-imp-cP*: [|cSumFp(p,x);In(y,p)|] ==> cP(y,x)
⟨proof⟩

theorem *bSumFp-and-Fp-imp-E*: bSumFp(p,x) --> (ALL t. (Fp(p,t) --> E(x,t)))
⟨proof⟩

theorem *bSumFp-and-bSumFp-and-Fp-imp-Me*: [|bSumFp(p,x);bSumFp(p,y);Fp(p,t)|] ==> Me(x,y,t)
⟨proof⟩

theorem *bSumFp-and-cSumFp-and-Subseteq-imp-cP*: [|bSumFp(p,x);cSumFp(q,y);Subseteq(p,q)|] ==> cP(x,y)
⟨proof⟩

theorem *cSumFp-and-In-imp-pSumFp*: [|cSumFp(p,x);In(x,p)|] ==> pSumFp(p,x)
⟨proof⟩

theorem *pSumFp-imp-Fp-iff-SumFp*: pSumFp(p,x) ==> (ALL t. (Fp(p,t) <-> SumFp(p,x,t)))
⟨proof⟩

theorem *pSumFp-imp-E-iff-SumFp*: pSumFp(p,x) ==> (ALL t. (E(x,t) <->

SumFp(*p*,*x*,*t*)))
⟨*proof*⟩

theorem *pSumFp-and-pSumFp-imp-E-iff-E*: [|*pSumFp*(*p*,*x*);*pSumFp*(*p*,*y*)|] ==>
(*ALL t*. (*E*(*x*,*t*) <−> *E*(*y*,*t*)))
⟨*proof*⟩

theorem *pSumFp-and-pSumFp-imp-E-iff-ME*: [|*pSumFp*(*p*,*x*);*pSumFp*(*p*,*y*)|] ==>
(*ALL t*. (*E*(*x*,*t*) <−> *Me*(*x*,*y*,*t*)))
⟨*proof*⟩

end
theory *Universals* **imports** *FOL*

begin

typedecl *Un*

arities *Un* :: *term*

consts

IsA :: *Un* => *Un* => *o*
IsAPr :: *Un* => *Un* => *o*
IsARoot :: *Un* => *o*
IsAI :: *Un* => *Un* => *o*
IsAO :: *Un* => *Un* => *o*

axioms

IsA-refl: (*ALL c*. (*IsA*(*c*,*c*)))
IsA-antisym: (*ALL c d*. (*IsA*(*c*,*d*) & *IsA*(*d*,*c*) −−> *c* = *d*))
IsA-trans: (*ALL c d e*. (*IsA*(*c*,*d*) & *IsA*(*d*,*e*) −−> *IsA*(*c*,*e*)))
IsA-wsp-IsAI: (*ALL c d*. (*IsAPr*(*c*,*d*) −−> (*EX e*. (*IsAPr*(*e*,*d*) & ~*IsAI*(*e*,*c*)))))

IsA-npo: *ALL c d*. (*IsAO*(*c*,*d*) −−> *IsAI*(*c*,*d*))
IsA-root: (*EX c*. *IsARoot*(*c*))

defs

IsAPr-def: *IsAPr*(*c*,*d*) == *IsA*(*c*,*d*) & ~*IsA*(*d*,*c*)
IsARoot-def: *IsARoot*(*d*) == (*ALL c*. *IsA*(*c*,*d*))
IsAI-def: *IsAI*(*c*,*d*) == *IsA*(*c*,*d*) | *IsA*(*d*,*c*)
IsAO-def: *IsAO*(*c*,*d*) == (*EX e*. (*IsA*(*e*,*c*) & *IsA*(*e*,*d*)))

lemma *IsA-antisym-rule* : [|*IsA*(*c*,*d*);*IsA*(*d*,*c*)|] ==> *c* = *d*

⟨*proof*⟩

lemma *IsA-trans-rule*: [|*IsA*(*c*,*d*);*IsA*(*d*,*e*)|] ==> *IsA*(*c*,*e*)
⟨*proof*⟩

lemma *IsA-wsp-IsAI-rule*: *IsAPr*(*c*,*d*) ==> (*EX e.* (*IsAPr*(*e*,*d*) & ~*IsAI*(*e*,*c*)))

⟨*proof*⟩

lemma *IsA-npo-rule*: [|*IsA*(*e*,*c*);*IsA*(*e*,*d*)|] ==> *IsAI*(*c*,*d*)
⟨*proof*⟩

lemma *IsA-npo-rule1*: *IsAO*(*c*,*d*) ==> *IsAI*(*c*,*d*)
⟨*proof*⟩

theorem *IsA-impl-Id-or-IsAPr*: *IsA*(*c*,*d*) ==> (*c*=*d* | *IsAPr*(*c*,*d*))
⟨*proof*⟩

theorem *IsARoot-unique*: [|*IsARoot*(*c*);*IsARoot*(*d*)|] ==> *c*=*d*
⟨*proof*⟩

theorem *IsAI-refl*: (*ALL c. IsAI*(*c*,*c*))
⟨*proof*⟩

theorem *IsA-imp-IsAI*: *IsA*(*c*,*d*) ==> *IsAI*(*c*,*d*)
⟨*proof*⟩

theorem *IsAPr-imp-IsA*: *IsAPr*(*c*,*d*) ==> *IsA*(*c*,*d*)
⟨*proof*⟩

theorem *IsAI-imp-IsAI-imp-IsA-rule*: (*ALL e.* (*IsAI*(*e*,*c*)-->*IsAI*(*e*,*d*))) ==> *IsA*(*c*,*d*)
⟨*proof*⟩

lemma *IsAI-imp-IsAI-imp-IsA*: (*ALL c d.* (*ALL e.* (*IsAI*(*e*,*c*)-->*IsAI*(*e*,*d*))) --> *IsA*(*c*,*d*))
⟨*proof*⟩

lemma *ltb3*: (*ALL e.* (*A*(*e*,*c*) & *B*(*e*,*d*))) ==> (*ALL e. A*(*e*,*c*)) & (*ALL e. B*(*e*,*d*))

⟨*proof*⟩

theorem *IsAI-iff-IsAI-iff-eq*: (*ALL c d*. (*ALL e*. (*IsAI*(*e*,*c*) <−> *IsAI*(*e*,*d*))) <−> *c*=*d*)
⟨*proof*⟩

theorem *IsA-wsp*: *IsAPr*(*c*,*d*) ==> (*EX e*. (*IsAPr*(*e*,*d*) & ~(*EX f*. *IsA*(*f*,*e*) & *IsA*(*f*,*c*))))
⟨*proof*⟩

theorem *IsA-and-IsAI-impl-IsAI*: [|*IsA*(*c*,*d*);*IsAI*(*c*,*e*)|] ==> *IsAI*(*d*,*e*)
⟨*proof*⟩

theorem *IsA-and-not-IsAI-impl-not-IsAI*: [|*IsA*(*c*,*d*);~*IsAI*(*d*,*e*)|] ==> ~*IsAI*(*c*,*e*)
⟨*proof*⟩

theorem *IsAI-impl-IsA-and-IsA*: *IsAI*(*c*,*d*) ==> *IsAO*(*c*,*d*)
⟨*proof*⟩

theorem *IsAI-iff-IsAO*: *ALL c d*. (*IsAI*(*c*,*d*) <−> *IsAO*(*c*,*d*))
⟨*proof*⟩

end
theory *Instantiation*

imports *TNEMO Universals*

begin

consts

Inst :: *Ob* => *Un* => *Ti* => *o*

axioms

Inst-IsA: (*ALL c d t x*. (*IsA*(*c*,*d*) −−> (*Inst*(*x*,*c*,*t*) −−> *Inst*(*x*,*d*,*t*))))
Inst-IsAI: (*ALL x c d t*. ((*Inst*(*x*,*c*,*t*) & *Inst*(*x*,*d*,*t*) −−> *IsAI*(*c*,*d*))))
Inst-E: (*ALL x c t*. (*Inst*(*x*,*c*,*t*) −−> *E*(*x*,*t*)))
Inst-Un: (*ALL c*. (*EX x t*. (*Inst*(*x*,*c*,*t*))))
Inst-Ob: (*ALL x t*. (*E*(*x*,*t*) −−> (*EX c*. (*Inst*(*x*,*c*,*t*)))))

lemma *Inst-IsA-rule*: *IsA*(*c*,*d*) ==> (*ALL x t*. (*Inst*(*x*,*c*,*t*) −−> *Inst*(*x*,*d*,*t*)))
⟨*proof*⟩

lemma *Inst-IsAI-rule*: [|*Inst*(*x*,*c*,*t*);*Inst*(*x*,*d*,*t*)|] ==> *IsAI*(*c*,*d*)
⟨*proof*⟩

lemma *Inst-Ob-rule*: (E(x,t) ==> (EX c. (Inst(x,c,t))))
⟨proof⟩

end
theory *ExtensionsOfUniversals*

imports *Instantiation Collections*

begin

consts

ExtCo :: Co => Un => Ti => o
ExtOb :: Ob => Un => Ti => o
DUn :: Un => o

axioms

Inst-impl-ExtOb-or-ExtCo: Inst(x,c,t) ==> (ExtOb(x,c,t) | (EX p. ExtCo(p,c,t)))

defs

ExtCo-def: ExtCo(p,c,t) == (ALL x. (In(x,p) <-> Inst(x,c,t)))
ExtOb-def: ExtOb(x,c,t) == (Inst(x,c,t) & (ALL y. (Inst(y,c,t) --> x=y)))
DUn-def: DUn(c) == (ALL t. (ALL p. ExtCo(p,c,t) --> DCo(p,t)))

theorem *DistinctInsts-impl-ExtCo*: [|Inst(x,c,t);Inst(y,c,t); (x~=y)|] ==> (EX p. ExtCo(p,c,t))
⟨proof⟩

theorem *ExtOb-unique*: [|ExtOb(x,c,t);ExtOb(y,c,t)|] ==> x=y
⟨proof⟩

theorem *ExtCo-unique*: [|ExtCo(p,c,t);ExtCo(q,c,t)|] ==> p = q
⟨proof⟩

theorem *ExtCo-Fp*: ExtCo(p,c,t) ==> Fp(p,t)
⟨proof⟩

theorem *ExtCo-impl-2members*: ExtCo(p,c,t) ==> (EX x y. (In(x,p) & In(y,p) & x ~= y))
⟨proof⟩

theorem *IsA-impl-Subseteq*: [|IsA(c,d);ExtCo(p,c,t);ExtCo(q,d,t)|] ==> Subseteq(p,q)

⟨*proof*⟩

theorem *ExtOb-impl-notExtCo*: *ExtOb*(*x*,*c*,*t*) ==> (~(*EX p*. *ExtCo*(*p*,*c*,*t*)))
⟨*proof*⟩

theorem *ExtCo-impl-notExtOb*: *ExtCo*(*p*,*c*,*t*) ==> (~(*EX x*. *ExtOb*(*x*,*c*,*t*)))
⟨*proof*⟩

theorem *NoExt-or-ExtOb-or-ExtDCo-impl-DUn*: (*ALL t*. ((~ (*EX x*. *Inst*(*x*,*c*,*t*))) | (*EX x*. *ExtOb*(*x*,*c*,*t*)) | (*EX p*. *ExtCo*(*p*,*c*,*t*) & *DCo*(*p*,*t*)))) ==> *DUn*(*c*)
⟨*proof*⟩

theorem *DUn-impl-NoExt-or-ExtOb-or-ExtDCo*: *DUn*(*c*) ==> (*ALL t*. ((~ (*EX x*. *Inst*(*x*,*c*,*t*))) | (*EX x*. *ExtOb*(*x*,*c*,*t*)) | (*EX p*. *ExtCo*(*p*,*c*,*t*) & *DCo*(*p*,*t*))))
⟨*proof*⟩

theorem *DUn-iff-NoExt-or-ExtOb-or-ExtDCo*: *DUn*(*c*) <−> (*ALL t*. ((~ (*EX x*. *Inst*(*x*,*c*,*t*))) | (*EX x*. *ExtOb*(*x*,*c*,*t*)) | (*EX p*. *ExtCo*(*p*,*c*,*t*) & *DCo*(*p*,*t*))))
⟨*proof*⟩

theorem *DUn-and-O-impl-Id*: [|*DUn*(*c*);*Inst*(*x*,*c*,*t*);*Inst*(*y*,*c*,*t*);*O*(*x*,*y*,*t*)|] ==> *x* = *y*
⟨*proof*⟩

end

theory *PartonomicInclusion*

imports *TNEMO Collections*

begin

consts

P1 :: *Co* => *Co* => *Ti* => *o*
P2 :: *Co* => *Co* => *Ti* => *o*
P12 :: *Co* => *Co* => *Ti* => *o*
DP1 :: *Co* => *Co* => *Ti* => *o*
DP2 :: *Co* => *Co* => *Ti* => *o*
DP12 :: *Co* => *Co* => *Ti* => *o*

defs

P1-def: *P1(p,q,t) == (ALL x. (In(x,p) --> (EX y. (In(y,q) & P(x,y,t)))))*
P2-def: *P2(p,q,t) == (ALL y. (In(y,q) --> (EX x. (In(x,p) & P(x,y,t)))))*
P12-def: *P12(p,q,t) == P1(p,q,t) & P2(p,q,t)*

DP1-def: *DP1(p,q,t) == P1(p,q,t) & DCo(p,t) & DCo(q,t)*
DP2-def: *DP2(p,q,t) == P2(p,q,t) & DCo(p,t) & DCo(q,t)*
DP12-def: *DP12(p,q,t) == DP1(p,q,t) & DP2(p,q,t)*

theorem *P1-imp-Fp-Pp*: *P1(p,q,t) ==> (Fp(p,t) & Pp(q,t))*
⟨*proof*⟩

theorem *P2-imp-Pp-Fp*: *P2(p,q,t) ==>(Pp(p,t) & Fp(q,t))*
⟨*proof*⟩

theorem *P12-imp-Pp-Fp*: *P12(p,q,t) ==> (Fp(p,t) & Fp(q,t))*
⟨*proof*⟩

theorem *Fp-iff-P1*: *(ALL p t. (Fp(p,t) <-> P1(p,p,t)))*
⟨*proof*⟩

theorem *P1-iff-P2*: *(ALL p t. (P1(p,p,t) <-> P2(p,p,t)))*
⟨*proof*⟩

theorem *P2-iff-P12*: *(ALL p t. (P2(p,p,t) <-> P12(p,p,t)))*
⟨*proof*⟩

theorem *P1-trans*: [|*P1(p,q,t)*;*P1(q,r,t)*|] ==> *P1(p,r,t)*
⟨*proof*⟩

theorem *P2-trans*: [|*P2(p,q,t)*;*P2(q,r,t)*|] ==> *P2(p,r,t)*
⟨*proof*⟩

theorem *P12-trans*: [|*P12(p,q,t)*;*P12(q,r,t)*|] ==> *P12(p,r,t)*
⟨*proof*⟩

theorem *DP1-trans*: [|*DP1(p,q,t)*;*DP1(q,r,t)*|] ==> *DP1(p,r,t)*
⟨*proof*⟩

theorem *DP2-trans*: [|*DP2(p,q,t)*;*DP2(q,r,t)*|] ==> *DP2(p,r,t)*
⟨*proof*⟩

theorem *DP12-trans*: [|*DP12*(*p*,*q*,*t*);*DP12*(*q*,*r*,*t*)|] ==> *DP12*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P1-imp-P1-1*: [|*Subseteq*(*r*,*p*);*P1*(*p*,*q*,*t*)|] ==> *P1*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P1-imp-P1-2*: [|*Subseteq*(*q*,*r*);*P1*(*p*,*q*,*t*)|] ==> *P1*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P2-imp-P2-1*: [|*Subseteq*(*r*,*q*);*P2*(*p*,*q*,*t*)|] ==> *P2*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P2-imp-P2-2*: [|*Subseteq*(*p*,*r*);*P2*(*p*,*q*,*t*)|] ==> *P2*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P12-imp-P1-1*: [|*Subseteq*(*r*,*p*);*P12*(*p*,*q*,*t*)|] ==> *P1*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P12-imp-P2-2*: [|*Subseteq*(*p*,*r*);*P12*(*p*,*q*,*t*)|] ==> *P2*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P12-imp-P2-1*: [|*Subseteq*(*r*,*q*);*P12*(*p*,*q*,*t*)|] ==> *P2*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-P12-imp-P1-2*: [|*Subseteq*(*q*,*r*);*P12*(*p*,*q*,*t*)|] ==> *P1*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP1-imp-DP1*: [|*Subseteq*(*r*,*p*);*DP1*(*p*,*q*,*t*)|] ==> *DP1*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP1-imp-P1*: [|*Subseteq*(*q*,*r*);*DP1*(*p*,*q*,*t*)|] ==> *P1*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP2-imp-DP2*: [|*Subseteq*(*r*,*q*);*DP2*(*p*,*q*,*t*)|] ==> *DP2*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP2-imp-P2*: [|*Subseteq*(*p*,*r*);*DP2*(*p*,*q*,*t*)|] ==> *P2*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP12-imp-DP1*: [|*Subseteq*(*r*,*p*);*DP12*(*p*,*q*,*t*)|] ==> *DP1*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP12-imp-P2*: [|*Subseteq*(*p*,*r*);*DP12*(*p*,*q*,*t*)|] ==> *P2*(*r*,*q*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP12-imp-DP2*: [|*Subseteq*(*r*,*q*);*DP12*(*p*,*q*,*t*)|] ==> *DP2*(*p*,*r*,*t*)
⟨*proof*⟩

theorem *Subseteq-and-DP12-imp-P1*: [|*Subseteq*(*q*,*r*);*DP12*(*p*,*q*,*t*)|] ==> *P1*(*p*,*r*,*t*)
⟨*proof*⟩

end

theory *UniversalParthood*

imports *ExtensionsOfUniversals PartonomicInclusion*

begin

consts

UPt1 :: *Un* => *Un* => *Ti* => *o*
UPt2 :: *Un* => *Un* => *Ti* => *o*
UPt12 :: *Un* => *Un* => *Ti* => *o*

UP1 :: *Un* => *Un* => *o*
UP2 :: *Un* => *Un* => *o*
UP12 :: *Un* => *Un* => *o*

defs

UPt1-def: *UPt1*(*c*,*d*,*t*) == (*ALL x*. (*Inst*(*x*,*c*,*t*) −−> (*EX y*. (*Inst*(*y*,*d*,*t*) & *P*(*x*,*y*,*t*)))))
UPt2-def: *UPt2*(*c*,*d*,*t*) == (*ALL y*. (*Inst*(*y*,*d*,*t*) −−> (*EX x*. (*Inst*(*x*,*c*,*t*) & *P*(*x*,*y*,*t*)))))
UPt12-def: *UPt12*(*c*,*d*,*t*) == *UPt1*(*c*,*d*,*t*) & *UPt2*(*c*,*d*,*t*)

UP1-def: *UP1*(*c*,*d*) == (*ALL t*. *UPt1*(*c*,*d*,*t*))
UP2-def: *UP2*(*c*,*d*) == (*ALL t*. *UPt2*(*c*,*d*,*t*))
UP12-def: *UP12*(*c*,*d*) == (*ALL t*. *UPt12*(*c*,*d*,*t*))

theorem *ExtCo-and-ExtCo*: [|*ExtCo*(*p*,*c*,*t*);*ExtCo*(*q*,*d*,*t*)|] ==> (*P1*(*p*,*q*,*t*) <−>
UPt1(*c*,*d*,*t*))

⟨*proof*⟩

theorem *ExtOb-and-ExtOb*: [|*ExtOb*(*x*,*c*,*t*);*ExtOb*(*y*,*d*,*t*)|] ==> (*P*(*x*,*y*,*t*) <-> *UPt1*(*c*,*d*,*t*))
⟨*proof*⟩

theorem *ExtCo-and-ExtOb*: [|*ExtCo*(*p*,*c*,*t*);*ExtOb*(*x*,*d*,*t*)|] ==> ((*EX y*. (*Inst*(*y*,*c*,*t*) & *P*(*y*,*x*,*t*)))) <-> *UPt2*(*c*,*d*,*t*)
⟨*proof*⟩

theorem *ExtOb-and-ExtCo*: [|*ExtOb*(*x*,*c*,*t*);*ExtCo*(*p*,*d*,*t*)|] ==> ((*EX y*. (*Inst*(*y*,*d*,*t*) & *P*(*x*,*y*,*t*)))) <-> *UPt1*(*c*,*d*,*t*)
⟨*proof*⟩

theorem *UPt1-refl*: (*ALL c t*. *UPt1*(*c*,*c*,*t*))
⟨*proof*⟩

theorem *UPt2-refl*: (*ALL c t*. *UPt2*(*c*,*c*,*t*))
⟨*proof*⟩

theorem *UPt12-refl*: (*ALL c t*. *UPt12*(*c*,*c*,*t*))
⟨*proof*⟩

theorem *UPt1-trans*: [|*UPt1*(*c*,*d*,*t*);*UPt1*(*d*,*e*,*t*)|]==> *UPt1*(*c*,*e*,*t*)
⟨*proof*⟩

theorem *UPt2-trans*: [|*UPt2*(*c*,*d*,*t*);*UPt2*(*d*,*e*,*t*)|]==> *UPt2*(*c*,*e*,*t*)
⟨*proof*⟩

theorem *UPt12-trans*: [|*UPt12*(*c*,*d*,*t*);*UPt12*(*d*,*e*,*t*)|]==> *UPt12*(*c*,*e*,*t*)
⟨*proof*⟩

theorem *UP1-iff*: (*ALL c d*. (*UP1*(*c*,*d*) <-> (*ALL t x*. (*Inst*(*x*,*c*,*t*) --> (*EX y*. (*Inst*(*y*,*d*,*t*) & *P*(*x*,*y*,*t*)))))))
⟨*proof*⟩

theorem *UP2-iff*: (*ALL c d*. (*UP2*(*c*,*d*) <-> (*ALL t y*. (*Inst*(*y*,*d*,*t*) --> (*EX x*. (*Inst*(*x*,*c*,*t*) & *P*(*x*,*y*,*t*)))))))
⟨*proof*⟩

theorem *UP1-refl*: (*ALL c*. *UP1*(*c*,*c*))
⟨*proof*⟩

theorem *UP2-refl*: (*ALL c. UP2*(*c*,*c*))
⟨*proof*⟩

theorem *UP12-refl*: (*ALL c. UP12*(*c*,*c*))
⟨*proof*⟩

theorem *UP1-trans*: [|*UP1*(*c*,*d*);*UP1*(*d*,*e*)|]==> *UP1*(*c*,*e*)
⟨*proof*⟩

theorem *UP2-trans*: [|*UP2*(*c*,*d*);*UP2*(*d*,*e*)|]==> *UP2*(*c*,*e*)
⟨*proof*⟩

theorem *UP12-trans*: [|*UP12*(*c*,*d*);*UP12*(*d*,*e*)|]==> *UP12*(*c*,*e*)
⟨*proof*⟩

theorem *UP12-impl-UP1-and-UP2*: *UP12*(*c*,*d*) ==> (*UP1*(*c*,*d*) & *UP2*(*c*,*d*))
⟨*proof*⟩

theorem *IsA-and-UP1-impl-UP1-1*: [|*IsA*(*e*,*c*);*UP1*(*c*,*d*)|] ==> *UP1*(*e*,*d*)
⟨*proof*⟩

theorem *IsA-and-UP1-impl-UP1-2*: [|*IsA*(*d*,*e*);*UP1*(*c*,*d*)|] ==> *UP1*(*c*,*e*)
⟨*proof*⟩

theorem *IsA-and-UP2-impl-UP2-1*: [|*IsA*(*e*,*d*);*UP2*(*c*,*d*)|] ==> *UP2*(*c*,*e*)
⟨*proof*⟩

theorem *IsA-and-UP2-impl-UP2-2*: [|*IsA*(*c*,*e*);*UP2*(*c*,*d*)|] ==> *UP2*(*e*,*d*)
⟨*proof*⟩

theorem *IsA-and-UP12-impl-UP1-1*: [|*IsA*(*e*,*c*);*UP12*(*c*,*d*)|] ==> *UP1*(*e*,*d*)
⟨*proof*⟩

theorem *IsA-and-UP12-impl-UP2-2*: [|*IsA*(*c*,*e*);*UP12*(*c*,*d*)|] ==> *UP2*(*e*,*d*)
⟨*proof*⟩

theorem *IsA-and-UP12-impl-UP2-1*: [|*IsA*(*e*,*d*);*UP12*(*c*,*d*)|] ==> *UP2*(*c*,*e*)
⟨*proof*⟩

theorem *IsA-and-UP12-impl-UP1-2*: [|*IsA*(*d*,*e*);*UP12*(*c*,*d*)|] ==> *UP1*(*c*,*e*)
⟨*proof*⟩

theorem *IsA-imp-UP1*: *IsA*(*c*,*d*) ==> *UP1*(*c*,*d*)
⟨*proof*⟩

end

theory *BFO*

imports *QDistR Adjacency TMTL SumsPartitions UniversalParthood*

begin

end