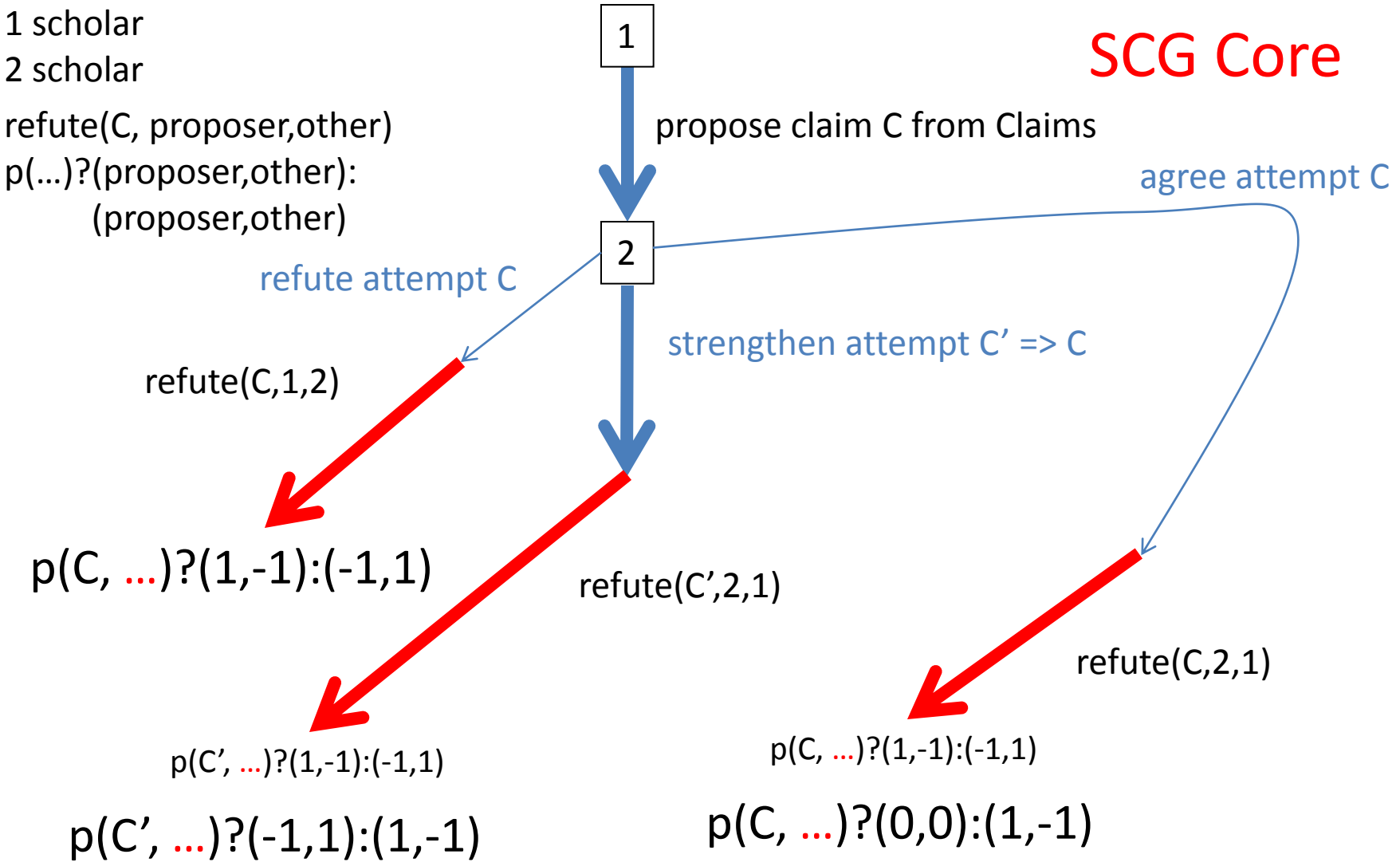
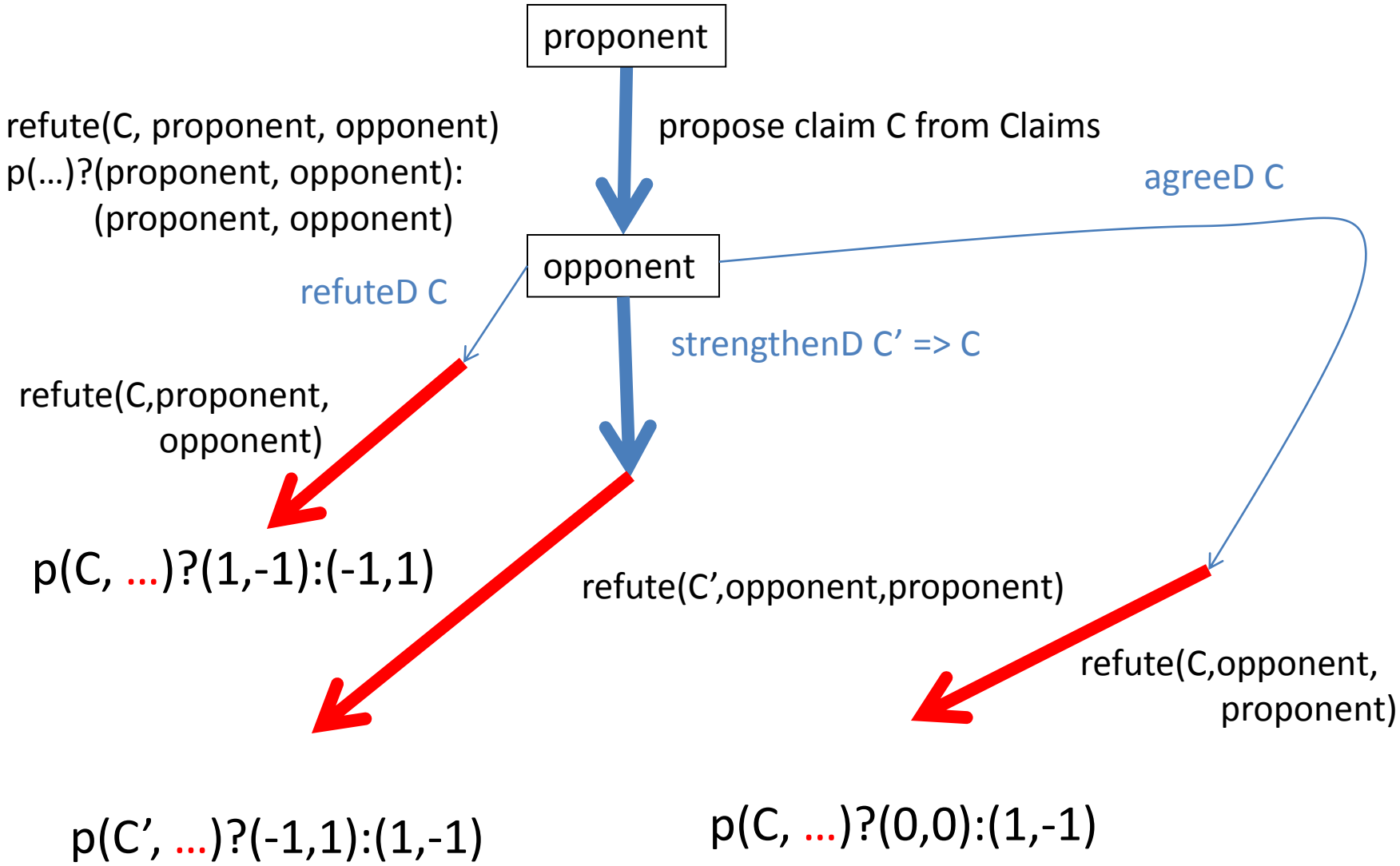
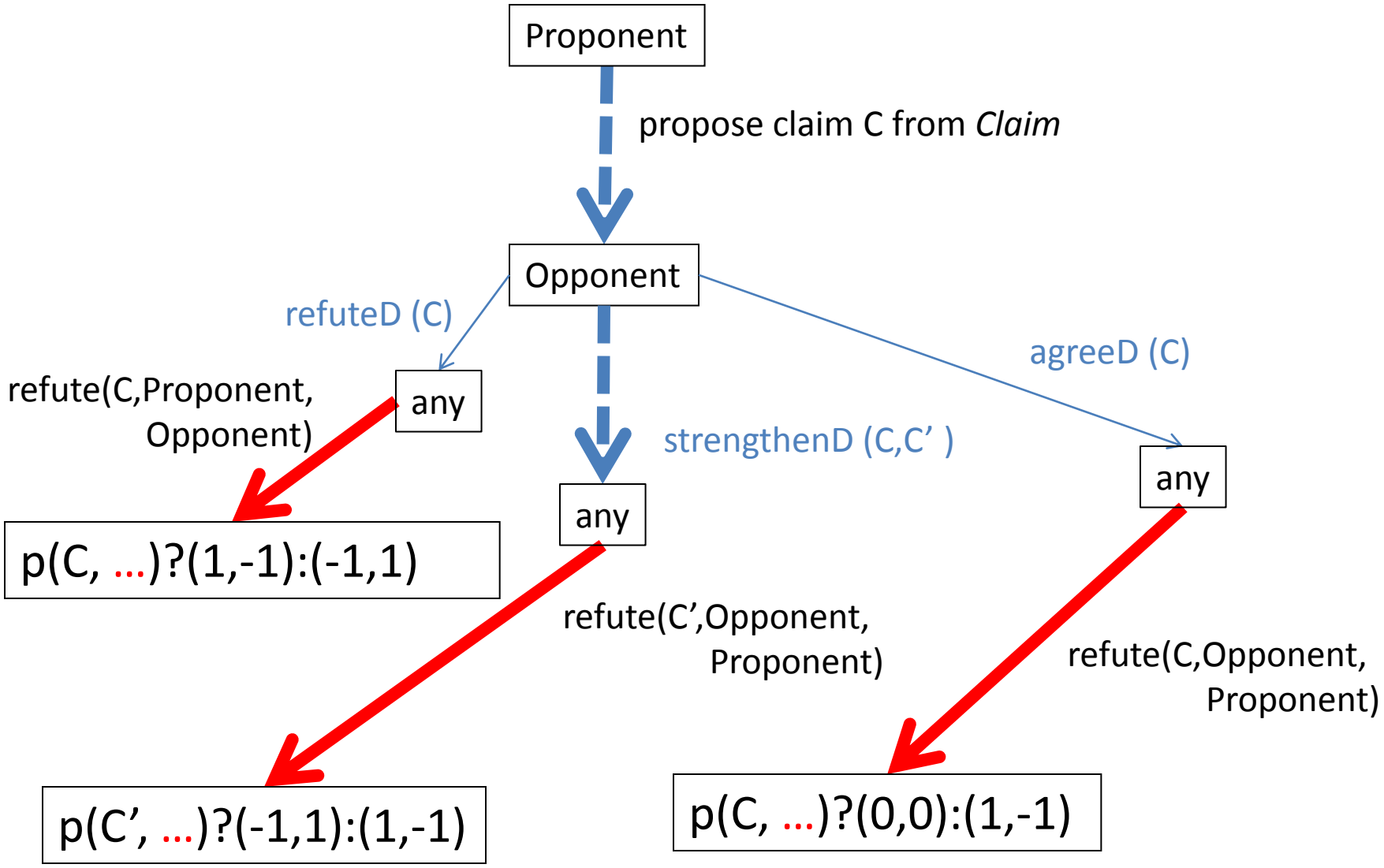
For Paper

1 scholar
2 scholar

refute(C, proposer,other)
p(…)?(proposer,other):
(proposer,other)




proponent
refute(C, proponent, opponent)
p(…)?(proponent, opponent):
(proponent, opponent)
propose claim C from Claims
agreeD C
opponent
refuteD C
strengthenD C' => C
refute(C,proponent,
opponent)
p(C, …)?(1,-1):(-1,1)
refute(C',opponent,proponent)
refute(C,opponent,
proponent)
p(C', …)?(-1,1):(1,-1)
p(C, …)?(0,0):(1,-1)


Proponent
propose claim C from *Claim*
Opponent
refuteD (C)
agreeD (C)
strengthenD (C,C')
refute(C,Proponent, Opponent)
any
any
any
p(C, ...)?(1,-1):(-1,1)
refute(C',Opponent, Proponent)
refute(C,Opponent, Proponent)
p(C', ...)?(-1,1):(1,-1)
p(C, ...)?(0,0):(1,-1)

Structures of SCG

Domain with name
- **Instance**
- **Solution**
- **valid(i: Instance, s: Solution**
- **quality(i: Instance, s: Solution**

Lab with name
- **d: Domain**
- **claim parameter definitions**
- **instance set predicate**
- **refutation predicate**
- **protocol**
- **stronger(c1,c2: Claim)**
- **distance(c1,c2: Claim)**

Claim with name
- **proponent: Scholar**
- **lab : Lab**
- **claim parameter values**

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: calculus problem

SaddlePoint
Instance = [0,1]
Solution = [0,1]
valid(i,s) = true
quality(i,s) = $i*s + (1-i)*(1-s^2)$

SaddlePointLab
SaddlePoint
q: [0,1]
true
quality(i[0],s[1])> =q
O:i[0], P:s[1] of i[0]
c1.q>c2.q
c1.q-c2.q

SPLClaim(
Alice,
SaddlePointLab,
0.6)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: worst-case of algorithm

GaleShapley (GS)
Nat
Preferences
valid(i,s) = s syntactically correct
quality(i,s) = GS iterations for s and i

GaleShapleyWorstCaseLab
GaleShapley
n:Nat, q: Nat
instanceSetP(i,n)=(i=n) //singleton
quality(i[0],s[1])>= q
O:i[0], P:s[1] of i[0]
c1.q>c2.q
c1.q-c2.q

GSWCLClaim(
Alice,
GaleShapleyWorstCaseLab,
10, 30)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: MaxSat

Satisfiability
CNF
Assignment
valid(i,s) = all variables in i assigned once
quality(i,s) = fraction of satisfied clauses in i

MaxSatLab
Satisfiability
q: [0,1], k: Nat (clause length)
instanceSetP(i,k)=clauses in i have length >=k
quality(i[0],s[1])> =q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

SJClaim(
Alice,
MaxSatLab,
$1-(1/2^3)$, 3)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: Boolean GeneralizedMaxSat = BMaxCSP

BooleanCSP
Sequence of Boolean constraints
Assignment
valid(i,s) = all variables in i assigned once
quality(i,s) =fraction of satisfied constraints in i

BooleanMaxCSPLab
BooleanCSP
q: [0,1], r: {R1,R2,…}
instanceSetP(i,r)=constraints in i use only r
quality(i[0],s[1])> =q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

BMCLClaim(
Alice,
BooleanMaxCSPLab,
0.618, {R1,R2})

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate

refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: BooleanMaxCSPLocalGlobal

BooleanCSP
Sequence of Boolean constraints
Assignment
valid(i,s) = all variables in i assigned once
quality(i,s) =fraction of satisfied constraints in i

BooleanMaxCSPLab
BooleanCSP
q: [0,1], r: {R1,R2,…}, k:Nat
instanceSetP(i,r,k)=(constraints in i use only r)
and any k constraints are satisfiable
quality(i[0],s[1])> =q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

BMCLClaim(
Alice,
BooleanMaxCSPLab,
0.618, {R1,R2,R3,R4})

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: Solar Cells

SolarCells
RawMaterials
Product
valid(i,s) = only raw materials used
quality(i,s) = energy efficiency

SollarCellsLab
SollarCells
q: [0,1], k: Nat (raw material parameter)
instanceSetP(i,k)= …
quality(i[0],s[1])> q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

SCLClaim(
Alice,
SolarCellsLab,
0.7, 3)

expression in a,d,n using multiplication, addition and division.
To simplify, replace (1+2+ … +n) by n*(n+1)/2.

$$\sum_{k=1}^{n} a + dk = (a + d) + (a + 2d) + (a + 3d) + \cdots + (a + nd) = na + (1 + 2 + 3 \cdots + n)d$$

Structures of SCG

Domain with name
- **Instance**
- **Solution**
- **valid(i: Instance, s: Solution**
- **quality(i: Instance, s: Solution**

Lab with name
- **d: Domain**
- **claim parameter definitions**
- **instance set predicate**
- **refutation predicate**
- **protocol**
- **stronger(c1,c2: Claim)**
- **distance(c1,c2: Claim)**

Claim with name
- **proponent: Scholar**
- **lab : Lab**
- **claim parameter values**

Example: Arithmetic Sequences Sum

ArithmeticSequences2
- triple a,d,n: Nat
- expression in a,d,n
- valid(i,s) = s uses +,*,/ and vars in i
- quality(i,s) = 1 if s is correct

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: Arithmetic Sequences Sum

ArithmeticSequences
expression in a,d,n: Nat uses +,*,/
assignment to a,d,n
valid(i,s) = i gives correct sum for s: relaxed
quality(i,s) = 1 iff valid(i,s) : strict

ArithmeticSequencesLab
ArithmeticSequences
none
singleton
quality(i[0],s[1])=1 true
O:i[0], P:s[1] of s[0]
false
0

ASLClaim(
Alice,
ArithmeticSequencesLab,
)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: Arithmetic Sequences Sum

ArithmeticSequencesInduction
sum[k=1..n] 2+3k = 2n+3(n(n+1))/2
sequence of steps: induction proof
valid(i,s) = proof s is correct induction proof
quality(i,s) = 1 iff valid(i,s)

ArithmeticSequencesInductionLab
ArithmeticSequencesInduction
equation
singleton
quality(i[0],s[1])=1
O:i[0], P:s[1] of s[0]
false
0

ASLClaim3(
Alice,
ArithmeticSequencesInductionLab,
sum[k=1..n] 2+3k = 2n+3(n(n+1))/2)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: HighestSafeRung

HighestSafeRung
pair(n,k)
decision tree
valid(i,s) = s is correct for (n,k)
quality(i,s) = depth(s)

HighestSafeRungLab
HighestSafeRung
n,k,q
singleton
quality(i[0],s[1])<=q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

HSRClaim(
Alice,
HighestSafeRungLab,
25,2,5)

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: LeafCovering

LeafCovering
Set of trees. Set M=subset of GCP of trees.
witness (leaf in GCP) of non-coverage by M
valid(i,s) = s is correct for i
quality(i,s) = unused

Structures of SCG

Domain with name
Instance
Solution
valid(i: Instance, s: Solution
quality(i: Instance, s: Solution

Lab with name
d: Domain
claim parameter definitions
instance set predicate
refutation predicate
protocol
stronger(c1,c2: Claim)
distance(c1,c2: Claim)

Claim with name
proponent: Scholar
lab : Lab
claim parameter values

Example: LeafCovering

LeafCovering
LeafCoveringProblem : Set of trees. Set M=subs
Program
valid(i,s) = s is correct for i
quality(i,s) = unused

LeafCoveringLab
LeafCovering
m: Nat (size of M)
instanceSetP(i,m)= |i.M|=m
quality(i[0],s[1])<=q
O:i[0], P:s[1] of s[0]
c1.q>c2.q
c1.q-c2.q

HSRClaim(
Alice,
HighestSafeRungLab,
25,2,5)