SCG Example Labs

Ahmed Abdelmeged
Karl Lieberherr
Structures of SCG

• These examples are to be read with the SCG paper as background.

• The best way to represent Domain, Lab and Claim is to have Domain and Lab as top-level classes and Claim nested inside Lab. Lab has a Domain as field to give all claims access to Domain functionality.

• Instance and Solution are nested inside Domain.
Structures of SCG

• We use a Java-like syntax but the goal is to use only one or two lines per item for those simple introductory labs.
Structures of SCG

Domain
Instance
Solution
valid(i: Instance, s: Solution)
quality(i: Instance, s: Solution)

Lab
d : Domain
proto: Protocol
Claim
claim parameters
isp(i:d.Instance)
p(I:d.Instance[],S:d.Solution[])
stronger(c2: Claim)
distance(c2: Claim)
Calculus Lab

• Have your students mastered calculus (minimizing and maximizing functions)?
• The next lab shows a lab to test their skills.
Structures of SCG

<table>
<thead>
<tr>
<th>Domain</th>
<th>Instance</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>valid(i: Instance, s: Solution)</td>
<td>quality(i: Instance, s: Solution)</td>
</tr>
</tbody>
</table>

Lab

- d: Domain
- proto: Protocol
- Claim
  - claim parameters
  - isp(i:d.Instance)
  - p(I:d.Instance[],S:d.Solution[])
  - stronger(c2: Claim)
  - distance(c2: Claim)

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
  - O:I[0], P:S[1] of I[0]
- SaddlePointLabClaim
  - q: [0,1]
  - isp(i)=true
  - p(I,S)=d.quality(I[0],S[1]) > =q
  - stronger(c2) = q>c2.q
  - distance(c2) = q-c2.q

new SaddlePointLab.Claim(q=0.6)
Programming an Algorithm

• Have your students understood the Gale-Shapley algorithm?

• Next come two labs where they can demonstrate their skills through their avatar in a full-round-robin tournament.
  – In the first lab they test each other’s programs to see whether they match each other’s best solution.
  – In the second lab, they find worst-case inputs.
Structures of SCG

**Domain**
- Instance
- Solution

**Solution**

- `valid(i: Instance, s: Solution)`
- `quality(i: Instance, s: Solution)`

---

Example: GS algorithm

**GaleShapleyBasic**
- Instance = Preferences
- Solution = Assignment

- `valid(i, s) = s is syntactically correct for i`
- `quality(i, s) = s is semantically correct for i`
  - 1 if true, 0 if false.

---

**GaleShapleyBasicLab**

**Claim**
- `d: Domain`
- `proto: Protocol`
- `Claim`
- `claim parameter definitions`
- `isp(i: d.Instance)`
- `p(I: d.Instance[], S: d.Solution[])`
- `stronger(c2: Claim)`
- `distance(c2: Claim)`

**NEW GSAtLeastAsGoodAsYouClaim()**
Structures of SCG

<table>
<thead>
<tr>
<th>Domain</th>
<th>Instance</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

valid(i: Instance, s: Solution)
quality(i: Instance, s: Solution)

Example: Worst-Case of GS algorithm

GaleShapley (GS)
Instance = Nat //number of people
Solution = Preferences
valid(i,s) = s is syntactically correct for i
quality(i,s) = GS iterations for s and i

Lab
d: Domain
proto: Protocol
Claim
claim parameters
isp(i:d.Instance)
p(I:d.Instance[],S:d.Solution[])
stronger(c2: Claim)
distance(c2: Claim)

GaleShapleyWorstCaseLab
GaleShapley
O:I[0], P:S[1] of I[0]
GSWCLClaim
n:Nat, q:Nat
isp(i)=(i=n) //singleton
p(I,S)=d.quality(I[0],S[1])>= q
stronger(c2)=this.q>c2.q
distance(c2)=this.q-c2.q

new GSWCLClaim(n=10,q=30)
Maximum Satisfiability

• The next lab is about a paper by David Johnson in the 1970’s which is covered now in some algorithm text books, like Kleinberg and Tardos.

• The following MaxSat lab covers several skills, such as working with randomized algorithms and then derandomizing them.
Structures of SCG

**Domain**
- Instance
- Solution

**Solution**
- valid(i: Instance, s: Solution)
- quality(i: Instance, s: Solution)

**Lab**
- d: Domain
- proto: Protocol
- Claim
  - claim parameters
  - isp(i:d.Instance)
  - p(I:d.Instance[], S:d.Solution[])
  - stronger(c2: Claim)
  - distance(c2: Claim)

**Example: MaxSat**

**Satisfiability**
- CNF
- Assignment

**MaxSatLab**
- Satisfiability
  - O:I[0], P:S[1] of S[0]
- MSLClaim
  - q:[0,1], k:Nat (clause length)
  - isp(i)=clauses in i have length >=k
  - p(I,S)=d.quality(I[0], S[1]) >= q
  - stronger(c2)=q > c2.q
  - distance(c2)=q - c2.q

**new MSLClaim(q=1-(1/2^3), k=3)**
Generalized MaxSat

• The next lab is based on a paper by Lieberherr and Specker (2012).
Structures of SCG

**Domain**
- Instance
- Solution

**Solution**
- valid\((i: \text{Instance, s: Solution})\)
- quality\((i: \text{Instance, s: Solution})\)

**Example:** Boolean GeneralizedMaxSat = BMaxCSP

**BooleanCSP**
- Sequence of Boolean constraints
- Assignment
- valid\((i,s) = \text{all variables in i assigned once}\)
- quality\((i,s) = \text{fraction of sat. constraints in i}\)

**Lab**
- D: Domain
- Proto: Protocol
- Claim
  - Claim parameters
  - isp\((i:d.\text{Instance})\)
  - p\((I:d.\text{Instance}[], S:d.\text{Solution}[])\)
  - stronger\((c2: \text{Claim})\)
  - distance\((c2: \text{Claim})\)

**BooleanMaxCSPLab**
- BooleanCSP
- O:\([0], P:\([1]\) of \([0]\)
- GenBooleanMaxSatClaim
  - q:\([0,1]\), r:\(\{R1,R2,...\}\)
  - isp\((i) = \text{constraints in i use only r}\)
  - p\((I,S) = d.\text{quality}(I[0], S[1]) \geq q\)
  - stronger\((c2) = q > c2.q\)
  - distance\((c2) = q - c2.q\)

new GenBooleanMaxSatClaim\((q=0.618, r=\{R1,R2}\))
Local to Global

• The next lab is based on several papers inspired by a JACM paper by Lieberherr and Specker in 1981.

• The lab is about studying how local properties of a conjunctive normal form translate into global properties.
Structures of SCG

Domain
Instance
Solution

valid(i: Instance, s: Solution)
quality(i: Instance, s: Solution)

Example: BooleanMaxCSPLocalGlobal

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

Lab

d: Domain
proto: Protocol
Claim
claim parameters
isp(i:d.Instance)

p(I:d.Instance[],S:d.Solution[])
stronger(c2: Claim)
distance(c2: Claim)

new BMCLClaim(q=0.618, r={R1,R2,R3,R4}, k=2)
Manufacturing Lab

• The next lab is about an efficient manufacturing problem where raw materials are turned into a product.

• The lab is underspecified in that the details about the isp function are missing.
Structures of SCG

Domain
Instance
Solution
valid(i: Instance, s: Solution)
quality(i: Instance, s: Solution)

Example: Solar Cells

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

Lab
    d: Domain
    proto: Protocol
    Claim
        claim parameters
        isp(i:d.Instance)
        p(l:d.Instance[],S:d.Solution[])
        stronger(c2: Claim)
        distance(c2: Claim)

SollarCellsLab
    SollarCells
    O:I[0], P:S[1] of I[0]
    SCLClaim
        q:[0,1], k:Nat (raw material parameter)
        isp(i)= ... 
        p(l,S)=d.quality(I[0],S[1])> q
        stronger(c2)=q>c2.q
        distance(c2)=q-c2.q

new SCLClaim(q=0.7,k=3)
Lab Reductions

• With the next example we show the usefulness of lab reductions. A lab L1 reduces to a lab L2 (L1 < L2) if a defense strategy for the claims in L2 guarantees a defense strategy for the claims in L1. Ideally, the claims in L2 are simpler.

• L1 reduces to L2 if we can use a black box for L2 to solve L1. The black box makes all perfect decisions, including claims it can defend.
Lab Reductions

• A mapping from L1 to L2 is a computable function f Domain Claim such that for any
  – L1.Domain L2.Domain
  – L1.Claim L2.Claim
  – propose
  – oppose/agree
  – provideInstance
  – solveInstance
  – refute
expression in a, d, n using multiplication, addition and division.
To simplify, replace \((1+2+ \ldots + n)\) 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

**Solution valid**
\(\text{valid}(i: \text{Instance}, s: \text{Solution})\)

**Solution quality**
\(\text{quality}(i: \text{Instance}, s: \text{Solution})\)

---

**Example: Arithmetic Sequences Sum**

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

---

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

---

**Claim with name**
- lab : Lab
- claim parameter values
Structures of SCG

**Domain with name**
- Instance
- Solution

**Instance**

valid(i: Instance, s: Solution)

**Solution**

quality(i: Instance, s: Solution)

---

**Lab with name**

- d: Domain
- claim parameter definitions
- instance set predicate
- refutation predicate
- protocol

**claim parameter definitions**

- stronger(c1,c2: Claim)
- distance(c1,c2: Claim)

---

**Claim with name**

- 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) = s assigns all 3 variables
- quality(i,s) = 1 iff i gives correct sum for s

---

- ASLClaim(
  - ArithmeticSequencesLab,
) //none
Structures of SCG

**Domain with name**
- Instance
- Solution

**valid**\(i: \text{Instance}, s: \text{Solution}\)

**quality**\(i: \text{Instance}, s: \text{Solution}\)

**Lab with name**
- d: Domain
- claim parameter definitions
- instance set predicate
- refutation predicate
- protocol

**stronger**\(c1,c2: \text{Claim}\)

**distance**\(c1,c2: \text{Claim}\)

Example: Arithmetic Sequences Sum

**ArithmeticSequencesInduction**
- sum\([k=1..n] 2+3k = 2n+3(n(n+1))/2\)
- sequence of steps: induction proof

**valid**\((i,s) = \text{proof s is syntactically correct}\)

**quality**\((i,s) = 1 \text{ iff proof is correct}\)

**Lab with name**
- ArithmeticSequencesInductionLab
- ArithmeticSequencesInduction
- equation
- singleton

**quality**\((i[0],s[1])=1\)

**O:i[0], P:s[1] of s[0]**

**false**

0

**Claim with name**
- ASLClaim3(\(\text{Alice,}\)
- 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
  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)

HSRClaim(
  HighestSafeRungLab,
  n->25,k->2,k->5)

lower quality is better
**Structures of SCG**

**Domain with name**
- Instance
- Solution

**valid(i: Instance, s: Solution)**
**quality(i: Instance, s: Solution)**

**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

**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: LeafCovering

LeafCovering
LeafCoveringProblem : Set of trees. Set M=subset of GCP of trees.
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

HSRCClaim(
  Alice,
  HighestSafeRungLab,
  25,2,5)
<table>
<thead>
<tr>
<th>claim</th>
<th>action</th>
<th>outcome</th>
<th>P</th>
<th>O</th>
<th>cB</th>
<th>aB</th>
<th>oB</th>
<th>np</th>
</tr>
</thead>
<tbody>
<tr>
<td>P not per</td>
<td>F *</td>
<td>a</td>
<td>sO</td>
<td>0</td>
<td>0</td>
<td>P</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>T</td>
<td>a</td>
<td>sO</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>O not per</td>
<td>F *</td>
<td>d</td>
<td>sP</td>
<td>1</td>
<td>-1</td>
<td>P</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>T</td>
<td>d</td>
<td>sP</td>
<td>1</td>
<td>-1</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>a</td>
<td>rP</td>
<td>-1</td>
<td>-1</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>O not per</td>
<td>T *</td>
<td>a</td>
<td>rP</td>
<td>-1</td>
<td>-1</td>
<td>-</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>F</td>
<td>d</td>
<td>rO</td>
<td>-1</td>
<td>1</td>
<td>P</td>
<td>-</td>
<td>P</td>
<td>-</td>
</tr>
<tr>
<td>P not per</td>
<td>T *</td>
<td>d</td>
<td>rO</td>
<td>-1</td>
<td>1</td>
<td>-</td>
<td>O</td>
<td>P</td>
</tr>
</tbody>
</table>
SCG Truth Table Interpretation

• no competition: P 0 and O 0 everywhere
• not fair: the player different from the “not perfect” player loses a point
<table>
<thead>
<tr>
<th>claim</th>
<th>dec</th>
<th>out</th>
<th>P</th>
<th>O</th>
<th>cB</th>
<th>aB</th>
<th>oB</th>
<th>Blame Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>F *</td>
<td>a</td>
<td>sO</td>
<td>0</td>
<td>0</td>
<td>P</td>
<td>O</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>a</td>
<td>sO</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>F *</td>
<td>d</td>
<td>sP</td>
<td>1</td>
<td>-1</td>
<td>P</td>
<td>-</td>
<td>O</td>
<td>O did not refute a claim it disputed</td>
</tr>
<tr>
<td>T</td>
<td>d</td>
<td>sP</td>
<td>1</td>
<td>-1</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>a</td>
<td>rP</td>
<td>1</td>
<td>-1</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>O failed to support a claim it agreed with</td>
</tr>
<tr>
<td>T *</td>
<td>a</td>
<td>rP</td>
<td>1</td>
<td>-1</td>
<td>-</td>
<td>-</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>d</td>
<td>rO</td>
<td>-1</td>
<td>1</td>
<td>P</td>
<td>-</td>
<td>P</td>
<td>P failed to support a claim it proposed</td>
</tr>
<tr>
<td>T *</td>
<td>d</td>
<td>rO</td>
<td>-1</td>
<td>1</td>
<td>-</td>
<td>O</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>claim</td>
<td>dec</td>
<td>out</td>
<td>P</td>
<td>O</td>
<td>cB</td>
<td>aB</td>
<td>oB</td>
<td>Blame Justification</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---------------------</td>
</tr>
<tr>
<td>F *</td>
<td>a</td>
<td>sO</td>
<td>paso</td>
<td>oaso</td>
<td>P</td>
<td>O</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>a</td>
<td>sO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>F *</td>
<td>d</td>
<td>sP</td>
<td>pdsp</td>
<td>odsp</td>
<td>P</td>
<td>-</td>
<td>O</td>
<td>O did not refute a claim it disputed</td>
</tr>
<tr>
<td>T</td>
<td>d</td>
<td>sP</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>a</td>
<td>rP</td>
<td>parp</td>
<td>oarp</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>O failed to support a claim it agreed with</td>
</tr>
<tr>
<td>T *</td>
<td>a</td>
<td>rP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>d</td>
<td>rO</td>
<td>pdro</td>
<td>odro</td>
<td>P</td>
<td>-</td>
<td>P</td>
<td>P failed to support a claim it proposed</td>
</tr>
<tr>
<td>T *</td>
<td>d</td>
<td>rO</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>