7 steps to adding a new solver

The implementation of Funlog
Multiple solvers

• Funlog incorporates several solvers
  – SAT solver
  – SMT solver
  – Mathematical programming solver

• Earlier version have also incorporated a max-Sat solver and a narrowing solver.
Modality

- A Funlog program has two kinds of modalities
  - Evaluation
  - Search

```plaintext
dim i10#Int = [0,1,2,3,4,5,6,7,8,9]
dim colors#String = ["Red","Blue","Green","Yellow"]

graph = [(1,2),(2,3),(3,4),(4,5),
        (5,1),(1,6),(2,7),(3,8),
        (4,9),(5,0),(6,8),(7,9),
        (8,0),(9,6),(0,7)]

edges = set #(i10,i10) graph
        -- color = toSet colors

pairs = [(1,"Red"),(2,"Red"),(3,"Blue")]
tim = set #(i10,colors) pairs

same r =
  let f(x,y,c) -> i10(x), i10(y), colors(c).
    f(x,y,c) <- $r(x,c), edges(x,y), $r(y,c).
  in f

justNodes r = ${ (n) <- $r(n,c)}

exists coloring: set #(i10,colors) none .. fullRel #(i10,colors)
where  none (same coloring)
    && full (justNodes coloring)
find  Many 4
    by  SAT
```
Overloading and Staging

- A primitive function or a user defined function can be used in both modalities.
- We think of these functions being overloaded (i.e. having two (or more) separate implementations.
- Internally we use staging to decide which implementation to use.
The Boolean Class

-- Something acts like a Boolean if it supports
-- these operations

class Show b => Boolean b where
  true :: b
  false :: b
  isTrue :: b -> Bool
  isFalse :: b -> Bool
  conj :: b -> b -> b     -- conjunction
  disj :: b -> b -> b     -- disjunction
  neg :: b -> b           -- negation
  imply :: b -> b -> b    -- implication
Number class

class NumLike t where
  liftI :: Int -> t
  liftR :: Rational -> t
  (+) :: t -> t -> t
  (*) :: t -> t -> t
Comparisons

class NumLike t, BoolLike b
    => Compare t b where
    (<=) :: t -> t -> b
    (==) :: t -> t -> b
Sets or Relations

class SetLike s where

  create:: Dim a -\to [a] -\to s a

  select:: (a -\to \text{Bool}) -\to s a -\to s a

  union:: s a -\to s a -\to s a

  proj3of3:: s (a,b,c) -\to s c
A solver

• Each solver answers questions over some kind of data.
• Each solver chooses a second representation type for the data it knows how to solve.
• The representation internally represents (part of) the input to a solver.
7 steps

1. Choose a representation type
2. Overload the primitive functions for the “real” type and the “representation” type
3. Describe how to initialize unknown existentially quantified data in the representation type.
4. Stage the search modality code
5. Execute the staged code to generate constraints
6. Format the constraints as input to a solver
7. Instantiate the solution back into a “real” piece of data.
SAT (Finite Set) Solvers. Step 1

data SAT =
    VarP Int
    | FalseP
    | TruthP
    | AndP SAT SAT
    | OrP SAT SAT
    | ImpliesP SAT SAT
    | NotP SAT
instance BoolLike SAT where
  true = TruthP
  false = FalseP
  (&&) = AndP
  liftB True = TruthP
  liftB False = False
instance BoolLike b => SetLike (BitVector b) where

create d xs = ...

select p (BV d xs) =
  BV d [(x, liftB (p x)) | (x,b) <- xs]
proj3of3 (BV (D3 _ _ d) xs) = ...
join (BV (D2 a b) xs) (BV (D2 _ c) ys) = ...

SAT (Finite Set) Solvers. Step 2
Example SAT problem

\( n = 3 \)
\[
\begin{align*}
\text{dim size} & \text{#Int} = [1..n] \\
\text{dim node} & \text{#Char} = "abcdefg"
\end{align*}
\]

\(-- cover(i, j, x, y, m, n) \quad \text{A block of size } i \times j, \text{ at point } (x, y), \text{ covers squares } (m, n)\)
\[
\text{cover} = \text{set } \#(\text{size, size, size, size, size, size}) \\
\quad [ \ (i, j, x, y, x+m, y+n) \\
\quad \ | \ i \leftarrow \text{size} \\
\quad \ , j \leftarrow \text{size} \\
\quad \ , x \leftarrow \text{size}, i+x \leq n+1 \\
\quad \ , y \leftarrow \text{size}, j+y \leq n+1 \\
\quad \ , m \leftarrow [0..i-1] \\
\quad \ , n \leftarrow [0..j-1] ]
\]

\(\text{rect} = \text{set } \#(\text{node, size, size}) \quad [('a',1,1),('b',1,2),('c',1,3),('d',2,1),('e',2,2),('f',3,2)]\)

\(\text{possible}(nm, i, j, x, y, m, n) \rightarrow \text{node}(nm), \text{size}(i), \text{size}(j), \text{size}(x), \text{size}(y), \text{size}(m), \text{size}(n).\)
\(\text{possible}(nm, i, j, x, y, m, n) \leftarrow \text{rect}(nm, i, j), \text{cover}(i, j, x, y, m, n).\)

\(\exists \text{ sol : set } \#(\text{node, size, size, size, size, size, size}) \quad \text{none .. possible}\)
\(\quad \text{where full } \{(m,n) \leftarrow \text{sol}(nm, i, j, x, y, m, n) \}\) \& \(\quad \text{-- every pair } (m,n) \text{ is covered}\)
\(\quad \quad \{ \text{sol}(nm, i, j, x, y, m, n) | \ nm \rightarrow (x,y) \} \) \& \(\quad \text{-- Each rect is used at most once}\)
\(\quad \quad \{ \text{sol}(nm, i, j, x, y, m, n) | (m,n) \rightarrow \text{nm} \} \) \(\quad \text{-- each pair is covered once}.\)

\(\text{find Abstract}\)
\(\text{by SAT}\)

\(\text{placement}(nm, x, y) \rightarrow \text{node}(nm), \text{size}(x), \text{size}(y).\)
\(\text{placement}(nm, x, y) \leftarrow \text{sol}(nm, i, j, x, y, m, n).\)
The concrete cover relation

\[(\text{Int#3}, \text{Int#3}, \text{Int#3}, \text{Int#3}, \text{Int#3}, \text{Int#3})\]
\[
\{(1,1,1,1,1,1), (1,1,1,2,1,2), (1,1,1,3,1,3), (1,1,2,1,2,1), \\
(1,1,2,2,1,2), (1,1,2,3,2,3), (1,1,3,1,3,1), (1,1,3,2,3,2), \\
(1,1,3,3,3,3), (1,2,1,1,1,1), (1,2,1,1,1,2), (1,2,1,2,1,2), \\
(1,2,1,2,2,2), (1,2,2,1,2,1), (1,2,2,1,2,2), (1,2,2,2,2,2), \\
(1,2,2,2,2,3), (1,2,3,1,3,1), (1,2,3,1,3,2), (1,2,3,2,3,2), \\
(1,2,3,2,3,3), (1,3,1,1,1,1), (1,3,1,1,1,2), (1,3,1,1,1,3), \\
(1,3,2,1,2,1), (1,3,2,1,2,2), (1,3,2,1,2,3), (1,3,3,1,3,1), \\
(1,3,3,1,3,2), (1,3,3,1,3,3), (2,1,1,1,1,1), (2,1,1,1,1,2), \\
(2,1,1,1,2,1), (2,1,1,2,2,2), (2,1,1,3,1,3), (2,1,1,3,2,3), \\
(2,1,2,1,2,1), (2,1,2,1,3,1), (2,1,2,2,2,2), (2,1,2,2,3,2)\}
possible(nm,i,j,x,y,m,n) <- rect(nm,i,j), cover(i,j,x,y,m,n).

rect = set #(node,size,size)
[('a',1,1),('b',1,2),('c',1,3),('d',2,1),('e',2,2),('f',3,2)]

exp> possible
(Char#7,Int#3,Int#3,Int#3,Int#3,Int#3,Int#3,Int#3)
{('a',1,1,1,1,1,1) ('a',1,1,1,2,1,2) ('a',1,1,1,3,1,3)
 ('a',1,1,2,1,2,1) ('a',1,1,2,2,2,2) ('a',1,1,2,3,2,3)
 ('a',1,1,3,1,3,1) ('a',1,1,3,2,3,2) ('a',1,1,3,3,3,3)
 ('b',1,2,1,1,1,1) ('b',1,2,1,1,1,2) ('b',1,2,1,2,1,2)
 ('b',1,2,1,2,1,3) ('b',1,2,2,1,2,1) ('b',1,2,2,1,2,2)
 ('b',1,2,2,2,2,2) ('b',1,2,2,2,2,3) ('b',1,2,3,1,3,1)
 ('b',1,2,3,1,3,2) ('b',1,2,3,2,3,2) ('b',1,2,3,2,3,3)
 ('c',1,3,1,1,1,1) ('c',1,3,1,1,1,2) ('c',1,3,1,1,1,3)
SAT (Finite Set) Step 3

exists sol : set #(node,size,size,size,size,size,size,size) none .. Possible

exp> sol
(Char#7,Int#3,Int#3,Int#3,Int#3,Int#3,Int#3)
{('a',1,1,1,1,1,1)=p1 ('a',1,1,1,2,1,2)=p2 ('a',1,1,1,3,1,3)=p3
 ('a',1,1,2,1,2,1)=p4 ('a',1,1,2,2,2,2)=p5 ('a',1,1,2,3,2,3)=p6
 ('a',1,1,3,1,3,1)=p7 ('a',1,1,3,2,3,2)=p8 ('a',1,1,3,3,3,3)=p9
 ('b',1,2,1,1,1,1)=p10 ('b',1,2,1,1,1,2)=p11 ('b',1,2,1,2,1,2)=p12
 ('b',1,2,1,2,1,3)=p13 ('b',1,2,2,1,2,1)=p14 ('b',1,2,2,1,2,2)=p15
 ('b',1,2,2,2,2,2)=p16 ('b',1,2,2,2,2,3)=p17 ('b',1,2,3,1,3,1)=p18
 ('b',1,2,3,1,3,2)=p19 ('b',1,2,3,2,3,2)=p20 ('b',1,2,3,2,3,3)=p21
 ('c',1,3,1,1,1,1)=p22 ('c',1,3,1,1,1,2)=p23 ('c',1,3,1,1,1,3)=p24
 ('c',1,3,2,1,2,1)=p25 ('c',1,3,2,1,2,2)=p26 ('c',1,3,2,1,2,3)=p27}
col r n = $( \{ k <- r(i,j,k), j =\text{size } n \} )$
row r n = $( \{ k <- r(i,j,k), i =\text{size } n \} )$
box r n = $( \{ k <- r(i,j,k), \text{square}(i,j,n) \} )$

exists grid : set $(\text{size},\text{size},\text{digit})$ none .. full
where and [ full (col grid i) | i <- size ] &&
      and [ full (row grid i) | i <- size ] &&
      and [ full (box grid i) | i <- size ] &&
$(\text{grid}(i,j,n),\text{grid}(i,j,m) \rightarrow n =\text{digit } m)$

find Many (setToArray grid)
by SAT
SAT (Finite Set) Step 5

where full \( \{ (m,n) \leftarrow \text{sol(nm,i,j,x,y,m,n)} \} \) \&\& --
every pair \((m,n)\) is covered

\( \{ \text{sol(nm,i,j,x,y,m,n)} \mid \text{nm} \rightarrow (x,y) \} \) \&\& --
Each rect is used at most once

\( \{ \text{sol(nm,i,j,x,y,m,n)} \mid (m,n) \rightarrow \text{nm} \} \) --
each pair is covered once.

Abstract where clause

\[(p1 \lor p10 \lor p22 \lor p31 \lor p43 \lor p59) \lor \]
\[(p2 \lor p11 \lor p12 \lor p23 \lor p33 \lor p44 \lor p47 \lor p60 \lor p65) \lor \]
\[(p3 \lor p13 \lor p24 \lor p35 \lor p48 \lor p66) \lor \]
\[(p4 \lor p14 \lor p25 \lor p32 \lor p37 \lor p45 \lor p51 \lor p61) \lor \ldots \]
SAT (Finite Set) Step 6

p cnf 70 559
1 10 22 31 43 59 0
2 11 12 23 33 44 47 60 65 0
3 13 24 35 48 66 0
4 14 25 32 37 45 51 61 0
5 15 16 26 34 39 46 49 52 55 62 67 0
6 17 27 36 41 50 56 68 0
7 18 28 38 53 63 0
8 19 20 29 40 54 57 64 69 0
9 21 30 42 58 70 0
-1 -2 0
-1 -3 0
-1 -4 0
...

...
SAT (Finite Set) Step 7

SAT
-1  2  -3  -4  -5  -6  -7  -8  -9  -10 -11 -12 -13  14  15 -16 -17 -18 -19 -
  20  -21  -22  -23  -24  -25  -26  -27  28  29  30 -31  -32  -33  -34  35  36  -
  69  -70  0

exp> sol
(Char#7,Int#3,Int#3,Int#3,Int#3,Int#3,Int#3)
{('a',1,1,1,1,1,1)=p1 ('a',1,1,1,2,1,2)=p2
 ('a',1,1,1,3,1,3)=p3
 ('a',1,1,2,1,2,1)=p4 ('a',1,1,2,2,2,2)=p5
 ('a',1,1,2,3,2,3)=p6
 ('a',1,1,3,1,3,1)=p7 ('a',1,1,3,2,3,2)=p8
 ('a',1,1,3,3,3,3)=p9
 ('b',1,2,1,1,1,1)=p10 ('b',1,2,1,1,1,2)=p11
 ('b',1,2,1,2,1,2)=p12

exp> sol
(Char#7,Int#3,Int#3,Int#3,Int#3,Int#3,Int#3)
{('a',1,1,1,2,1,2) ('b',1,2,2,1,2,1) ('b',1,2,2,1,2,2)
 ('c',1,3,3,1,3,1) ('c',1,3,3,1,3,2) ('c',1,3,3,1,3,3)
 ('d',2,1,1,3,1,3) ('d',2,1,1,3,2,3) ('e',2,2,1,1,1,1)}
mathematical programming problem

sum [] = 0
sum [x] = x
sum (x:xs) = x + sum xs

and [] = True
and [x] = x
and (x:xs) = x && and xs

--------------------------------------------

-- Production minimization problem

data Factory = A | B | C
data Store = NYC | ATL | LA

pairs = #(Factory,Store)

ship = array pairs [2,3,5,3,2,1,3,4,2]
sales = array Store [230,140,300]

exists prod: Array #(Factory,Store) Int
  where sum [ prod.(A,s) | s <- Store ] <= 150 &&
    and [ prod.(f,s) >= 0 | (f,s) <- pairs ] &&
    and [ sales.s == sum [prod.(f,s) | f <- Factory]
      | s <- Store ]

find Min sum [ prod.(f,s) * ship.(f,s)
  | (f,s) <- pairs ]

by IP
type PolyNom n = [(String, n)]

-- The meaning of an MExp is
-- a PolyNomial with an additive
-- constant. The polynomial may
-- have no polynomial terms

data MExp n = Term (PolyNom n) n

data Rel n
  = RANGE (PolyNom n) (Range n)
  | TAUT    -- True
  | UNSAT   -- False
deriving Eq
instance Num n => NumLike (Mexp n) where
  liftI n = Term [] n
  (+) = plusM (+)

plusM (Term [] a) (Term [] b) = Term [] (a+b)
plusM (Term [] a) (Term ys b) = Term ys (a+b)
plusM (Term xs a) (Term [] b) = Term xs (a+b)
plusM (Term xs a) (Term ys b) = Term (mergeP (+) xs ys) (a+b)

-- in a Sum, if the same indexed variables appears twice,
-- we add the coefficients
mergeP:: (Num n,Eq n) => (n -> n -> n)-> PolyNom n -> PolyNom n -> PolyNom n
mergeP f [] ys = ys
mergeP f xs [] = xs
mergeP f ((x,n):xs)((y,m):ys)=
  case compare x y of
    EQ -> case (f n m) of
      0 -> mergeP f xs ys
      i -> (x,i):mergeP f xs ys
    LT -> (x,n):
      mergeP f xs ((y,m):ys)
    GT -> (y,m):
      mergeP f ((x,n):xs) ys
instance Num n => BoolLike [Rel n] where
  true = Taut
  false = UnSat
  (&&) = AndP
  liftB True = Taut
  liftB False = UnSat

andM xs ys = help (sort xs) (sort ys)
  where
  help (UNSAT:_:ys) ys = [UNSAT]
  help xs (UNSAT:_:ys) = [UNSAT]
  help (TAUT:xs) ys = help xs ys
  help xs (TAUT:ys) = help xs ys
  help [] ys = ys
  help xs [] = xs
  help (RANGE x a:x:xs) (RANGE y b:y:ys)
    | x==y = RANGE x (intersectRange a b):help xs ys
  help (RANGE x a:x:xs)(ys@(RANGE y b:_))
    | x < y  = RANGE x a:(help xs ys)
  help (xs@(RANGE x a:_))(RANGE y b:y:ys)
    | x > y  = RANGE y b:(help xs ys)
instance (Num n) =>
  Compare (Mexp n) [Rel n] where
  (<=) = lteqM

lteqM (Term [] a) (Term [] b) =
  if (a <= b) then [TAUT] else [UNSAT]
lteqM (Term [] a) (Term xs b) =
  [RANGE xs (Range (LtEQ(a-b)) PlusInf)]
lteqM (Term xs a) (Term [] b) =
  [RANGE xs (Range MinusInf (LtEQ (b-a)))]
lteqM (Term xs a) (Term ys b) =
  [RANGE (mergeP (+) xs (negPoly ys))
   (Range MinusInf (LtEQ (b-a)))]
Math Prog Step 3

exists prod: Array #(Factory,Store) Int

Abstract values
prod =
    NYC ATL LA
    +-------------+
    A | `a | `b | `c |
    +-------------+
    B | `d | `e | `f |
    +-------------+
    C | `g | `h | `i |
    +-------------+

- The array is concrete, but the values are abstract
exists prod: Array #(Factory,Store) Int

where sum[ prod.(A,s) |
  | s <- Store ] <= (liftI 150) &&
and [ prod.(f,s) >= (liftI 0) |
  | (f,s) <- pairs ] &&
and [ sales.s == sum [prod.(f,s) |
  | f <- Factory] |
  | s <- Store ]
Math Prog Step 5

Abstract where clause

\[
\begin{align*}
0 & \leq [("a",1)] < +\text{Inf} , \\
-\text{Inf} & < [("a",1),("b",1),("c",1)] \leq 150 , \\
230 & \leq [("a",1),("d",1),("g",1)] \leq 230 , \\
0 & \leq [("b",1)] < +\text{Inf} , \\
140 & \leq [("b",1),("e",1),("h",1)] \leq 140 , \\
0 & \leq [("c",1)] < +\text{Inf} , \\
300 & \leq [("c",1),("f",1),("i",1)] \leq 300 , \\
0 & \leq [("d",1)] < +\text{Inf} , \\
0 & \leq [("e",1)] < +\text{Inf} , \\
0 & \leq [("f",1)] < +\text{Inf} , \\
0 & \leq [("g",1)] < +\text{Inf} , \\
0 & \leq [("h",1)] < +\text{Inf} , \\
0 & \leq [("i",1)] < +\text{Inf} \\
\end{align*}
\]
Math Prog Step 6

maximize (2a + 3b + 5c + 3d + 2e + 1f + 3g + 4h + 2i) where
0 <= (1a) < +Inf
-Inf < (1a + 1b + 1c) <= 150
230 <= (1a + 1d + 1g) <= 230
0 <= (1b) < +Inf
140 <= (1b + 1e + 1h) <= 140
0 <= (1c) < +Inf
300 <= (1c + 1f + 1i) <= 300
0 <= (1d) < +Inf
0 <= (1e) < +Inf
0 <= (1f) < +Inf
0 <= (1g) < +Inf
0 <= (1h) < +Inf
0 <= (1i) < +Inf

NAME          prod
ROWS
N  COST
L  R2
E  R3
E  R5
E  R7
COLUMNS
   COST         2
   R2           1
   R3           1
   R5           1
   R7           1
   COST         3
   R2           1
   R3           1
   R5           1
   R7           1
   COST         2
   R5           1
   R7           1
   COST         3
   R3           1
   R5           1
   R7           1
   COST         4
   R5           1
   R7           1
   COST         2
   R7           1
RHS
   R2     150
   R3     230
   R5     140
   R7     300
BOUNDS
   LO BND1   a     0
   LO BND1   b     0
   LO BND1   c     0
   LO BND1   d     0
   LO BND1   e     0
   LO BND1   f     0
   LO BND1   g     0
   LO BND1   h     0
   LO BND1   i     0
ENDATA
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>140</td>
<td>300</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

NYC | ATL | LA
+---+---+---+
| A | `a | `b | `c |
+---+---+---+
| B | `d | `e | `f |
+---+---+---+
| C | `g | `h | `i |
+---+---+---+