Diego Klabjan, Jun Ma, Robert Fourer – Northwestern University

ALGEBRAIC MODELING IN DATALOG

DATALOG

- Data query languages
  - SQL, Xquery
  - No procedural or declarative abilities
- Procedural and declarative languages
  - No data querying capabilities
- Best of both worlds

DATALOG

ALGEBRAIC MODELING

Specialized

- AMPL, AIMMS, OPL, GAMS
- MPL

Embedded in Languages

- Concert, Flops++ (C++)
- Pyomo, Poams (Python)

Embedded in Data Language

- SQL, PLAM (prolog), XQuery (OSmL)
- Datalog (LB)

SQL

- Modeling in SQL by Linderoth, Atamturk, Savelsbergh
- SQL mainly about querying
  - Not suited for algebraic modeling
- Everything stored in tables
  - Variables
  - Constraints
- Modeling non intuitive

MODELING IN DATALOG

- Powerful data capabilities
  - Superset of SQL
  - Data from database
  - Querying and loading
- Declarative language
  - Natural constructs
  - Intuitive
- Hardly any development effort

MODELING IN DATALOG

- Given values to decision variables
  - Easy to check feasibility
    - Clearly not optimality
  - Underlying logic programming in Datalog
  - No extra effort required
- Not the case for most other algebraic modeling languages

BASIC BUILDING BLOCKS (DIET PROBLEM)

- Index sets

```
NUTR(x), NUTR:name(x:n) -> string(n).
FOOD(x), FOOD:name(x:n) -> string(n).
```

- Parameters
  - Input data

```
amt[n, f] = a -> NUTR(n), FOOD(f), float[64](a), a >= 0.
nutrLow[n] = nL -> NUTR(n), float[64](nL), nL >= 0.
cost[f] = c -> FOOD(f), float[64](c), c >= 0.
```

- Variables

```
Buy[f] = b -> FOOD(f), float[64](b),b>=0.
```

PRODUCTION/TRANSPORTATION MODEL

- Multiple products (PROD), plants (ORIG) and destinations (DEST)
- Ship from plants to destinations
  - Transportation problem for each product

$$\sum_{j \in DEST} Trans_{i,j,p} = Make_{i,p} \quad i \in ORIG, p \in PROD$$

$$\sum_{i \in ORIG} Trans_{i,j,p} = demand_{j,p} \quad j \in DEST, p \in PROD$$

PRODUCTION/TRANSPORTATION MODEL

- Limited production capacity at each plant

$$\sum_{p \in PROD} (1/rate_{i,p} * Make_{i,p}) \leq avail_i \qquad i \in ORIG$$

- Objective to minimize production and transportation costs

$$\underset{x}{\text{minimize}} \sum_{i \in ORIG, p \in PROD} (makeCost_{i,p} * Make_{i,p}) + \sum_{i \in ORIG, j \in DEST, p \in PROD} (transCost_{i,j,p} * Trans_{i,j,p})$$

DATALOG MODEL

- Sets

$$PROD(p), PROD\!:\!name(p\!:\!n) -> \ string(n).$$

$$ORIG(o), ORIG\!:\!name(o\!:\!n) -> \ string(n).$$

$$DEST(d), DEST\!:\!name(d\!:\!n) -> \ string(n).$$

- Data

$$+PROD(p), +PROD\!:\!name(p\!:\!\text{SKU1}).$$

$$+PROD(p), +PROD\!:\!name(p\!:\!\text{SKU2}).$$

$$+PROD(p), +PROD\!:\!name(p\!:\!\text{SKU3}).$$

DATALOG MODEL

- Input parameters

$$avail[o] = av -> ORIG(o), float[32](av).$$

$$rate[o,p] = rv -> ORIG(o), PROD(p), float[32](rv).$$

$$demand[d,p] = dm -> DEST(d), PROD(p), float[32](dm).$$

$$makeCost[o,p] = mc -> ORIG(o), PROD(p), float[32](mc).$$

$$transCost[o,d,p] = tc -> ORIG(o), DEST(d), PROD(p), float[32](tc).$$

DATALOG MODEL

- Variables
  - How much to transport
  - How much to produce

$$Make[o,p] = mk -> ORIG(o), PROD(p), float[32](mk).$$

$$Trans[o,d,p] = tr -> ORIG(o), DEST(d), PROD(p), float[32](tr).$$

- Other variables
  - Integer replace “float” with “int”
  - Binary are integer with upper bound of 1

DATALOG MODEL

- Production availability
  - `sumTime` predicate captures the left-hand side
  - `agg` built-in aggregator
  - Constraint negated (stratification restrictions of Datalog)

$$sumTime[o] = st -> ORIG(o), float[32](st).$$

$$sumTime[o] = st <- agg \ll st = total(v) \gg$$

$$v = (1/rate[o,p]) * Make[o,p].$$

$$!(sumTime(o; t1), avail(o; t2), t1 > t2).$$

DATALOG MODEL

- Demand constraints

$$sumDemand[d,p] = v -> DEST(d), PROD(p), float[32](v).$$

$$sumDemand[d,p] = sd <- agg \ll sd = total(m) \gg$$

$$m = Trans[o,d,p].$$

$$!\,(sumDemand(d,p;dv), demand(d,p;mv), dv > mv).$$

DATALOG MODEL

- Supply constraints

$$sumSupply[o,p] \ = \ v -> \ ORIG(o), PROD(p), float[32](v).$$

$$sumSupply[o,p] = \ ss \ < - \, agg \ll ss = total(m) \gg$$

$$m \ = \ \ Trans[o,d,p].$$

$$!\,(sumSupply(o,p;sv), Make(o,p;mv), sv \ > \ mv).$$

DATALOG MODEL

- Objective function
  - Production cost

$$TotalMakeCost[] \;=\; tmc -> \; float[32](tmc).$$

$$TotalMakeCost[\quad] = tmc \;<- agg \ll tmc = total(v) \gg$$

$$v = make\_cost[o,p] \,*\, Make[o,p].$$

$$TotalTransCost[] \;=\; ttc -> \; float[32](ttc).$$

  - Transportation cost

$$TotalTransCost[\quad] = ttc \;<- agg \ll ttc = total(v) \gg$$

$$v = trans\_cost[o,d,p] \,*\, Trans[o,d,p].$$

TOTAL COST

- Sum of the two cost components

$$TotalCost[] = tc -> float[32](tc).$$

$$TotalCost[] = TotalMakeCost[] + TotalTransCost[].$$

ENHANCED MODELING CAPABILITIES

- The fleeting model
  - Assign fleets to flights
  - Modeled as a multi-commodity network flow problem
  - Network aspects
- Challenges
  - Nodes at each airports sorted based on the arrival/departure time in a circular fashion
  - Ordered and circular lists

FLIGHTS

- Specification of flights

```
Leg(l), Leg:name(l:n) -> string(n).
Leg:table(s1,t1,s2,t2,l) ->Station(s1),
  Time(t1), Station(s2), Time(t2), Leg(l).
Leg:table:dStation[l]=s1 ->Station(s1),
  Leg(l).
Leg:table:dTime[l]=t1 -> Time(t1), Leg(l).
Leg:table:aStation[l]=s2 ->Station(s2),
  Leg(l).
Leg:table:aTime[l]=t2 -> Time(t2), Leg(l).
```

NETWORK NODES

- For each station there is a timeline consisting of network nodes
  - Either arrival or departure at station

```
node(s,t) -> Station(s), Time(t).

node(s,t) <- Leg:table(s1,t1,_,_,_),(s=s1,
t=t1);Leg:table(_,_,s2,t2,_),(s=s2, t=t2).
```

ORDERED CIRCULAR LISTS

- Declare ‘next’ in the list
  - `Time` is an integer-like structure to capture times

```
node:nxt[s,t1] = t2 -> Station(s),
  Time(t1), Time(t2).
```

- Order

```
node:nxt[s,t1] = t2 <- node(s,t1),
  node(s,t2),
  (Time:datetime[t1]<Time:datetime[t2
  ],!anythingInBetween(s, t1,t2);
  node:frst[s]=t2, node:lst[s]=t1).
```

MISSING ASPECTS

- Piecewise linear functions
  - Model them explicitly
  - Unfortunately OS cannot handle them explicitly
    - LogicBlox needs to convert them into a linear mixed integer program
- Disjunctions
- Nonlinear functions
  - Long term goal