Towards Declarative Scripting
Combining CP and Analytics

CP 2015 Panel on CP and Analytics

Hassan Aït-Kaci

August 31, 2015
(rev. June 14, 2016)

Programme Avenir Lyon Saint-Etienne

UNIVERSITÉ DE LYON

UCLB

LIRIS

And so. . .

■

Purpose of this (short) presentation

■

Disclaimer: What this talk is not about…

It is:

- ▶ not a complete survey (of Analytics nor CP!)
- ▶ not a report of research/experimental results

This presentation's (hopeful) **objective**:

- ▶ high-level **glance at current trends** in the Analytics and CP landscape (the forest rather than the trees)
- ▶ speculate and **extrapolate some perspectives** therefrom (*i.e.*, what seems to be needed)

So what is this presentation about?

rapid overview of current trends in:

- the state of the art in **Statistical and Predictive Analytics**
- how it can leverage **CP technology**
- and ***vice versa!***

- ► **Analytics**—quick look at popular tools and trends in Statistics and OR
- ► **Synthesis**—CP meets Analytics
- ► **Prognosis**—declarative scripting?
- ► **Discussion**—recapitulation and requirements

And so. . .

Analytics

with

Statistical Analysis

Statistical Analytics

Used in **Decision Science** for:

computing/plotting probabilistic measures of data (moments—mean, variance, *etc.*), scatter-diagrams, trends, rates-of-change

drawing inference by correlation, regression, Bayes Law, *etc.*, in (discrete or continuous) autoregressive stochastic processes

- **not new**—has been around in varied forms since antiquity
  modern mathematical formulation due to Galton and Pearson (late 1800s–early 1900s)
- practical **computational tools for systematic data analysis**
  e.g., developed by the RAND Corporation since its inception
- **invaluable for data-driven decision-making**
  e.g., trend analysis forecasting (business and social sciences)

And so. . .

Analytics

Some popular statistical analytics tools

Some tools used for statistical analytics

- **some have been around for a while**; *e.g.*:
  - SPSS (at least since I was a grad student!)
  - SAS
  - Stata
  - S
- **new systems surf the Big Data wave**; *e.g.*:
  - R
  - Apache Spark (Hadoop)
  - RapidMiner (Radoop)
  - KMINE
  - Datameer's JSON Array Analytics

Most started as academic systems then went private (bought off or going corporate); most remain open-source

Current usage trends of statistical analytics tool

Top 10 tools by share of users: (KDNuggets, May 2015)

System	2015 % share	2014 % share	up down	2014–2015 % change
R	46.9	38.5	↗	+8.4
RapidMiner	31.5	44.2	↘	−12.7
SQL	30.9	25.3	↗	+5.6
Python	30.3	19.5	↗	+10.8
Excel	22.9	25.8	↘	−2.9
KNIME	20.0	15.0	↗	+5.0
Hadoop	18.4	12.7	↗	+5.7
Tableau	12.4	9.1	↗	+3.3
SAS	11.3	10.9	↗	+0.4
Spark	11.3	2.6	↗	+8.7

And so. . .

Analytics

with

CP/OR

Constraint Programming and Operations Research

Constrained Optimization and Operations Research

Formal models expressing (linear or quadratic) objective functions to min/max/imize subject to (linear) constraints over reals (LP/QP) or integers (IP) or both (MIP)

- **not new**—has been around for a while, but took off since Dantzig's Simplex algorithm (1939)
- **practical**—*e.g.*, as used by the RAND Corporation, esp. since Dantzig created its OR Dept (1940)
- invaluable **for strategic decision-making** (esp. military and business)
- **new (non-classical OR) CP techniques have emerged** (*e.g.*, arc consistency, *all-different*, SAT, BDDs, symmetry, *etc.*)

And so. . .

What is "scripting?"

Isn't it programming?

What is *scripting*?. . .

Isn't writing a JavaScript or Python script the same as writing a C, C++, C#, or Java program?

Or is it?

Yes and **no**:

- **Yes**: **scripting** is indeed a form of software programming in the sense that it is writing an executable coded specification of instructions; it **is the *glue* connecting application modules and actual data**.

- **No**: **scripting is not for high-performance static software development** producing well-honed blackboxes implementing the best-known *algorithms*.

What is *scripting*?. . .

- scripting programs are not statically compiled then executed: they are dynamically interpreted text-based source code (in particular, they can be put together as strings of text and executed on the fly)
- scripting specifies how to orchestrate several interacting static program apps into a coherent whole
- scripting may be seen as “light-weight” programming where the focus is not on the use of complex algorithms, but on a very large pool of tool libraries: both public, *etc.* and private.

Hence, scripting is more useful for specific purposes such as web-oriented visual-oriented dynamic jigsaw puzzle construction; akin to using pre-built construction blocks and/or building new ones to be (re-)used as libraries.

And so. . .

Some popular scripting tools

why they are more or less popular

(pros and cons)

Scripting tools

- Programming languages can be used for scripting; *e.g.*:
  - Java
  - Scala
  - Rust

But scripting is a specific kind of programming:

- Popular scripting tools; *e.g.*:
  - JavaScript
  - Python
  - IPython Notebook

Can be used in such systems as Apache Spark and Apache Flink

And so. . .

New Trends

Functional Scripting

(pros and cons)

Apache Spark:

Declarative notation for multithreaded MapReduce?

Quoting from their site, Spark. . . “***is a fast and general engine for large-scale data processing***.” It offers:

- **Speed**: run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk
- **Ease of use**: applications in Java, Scala, Python, R

  Word count using Spark’s Python API:

  ```
  text_file = spark.textFile("hdfs://...")
  text_file.flatMap(lambda line:  line.split())
           .map(lambda word:  (word, 1))
           .reduceByKey(lambda a, b:  a+b)
  ```
- **Generality**: complex analytics on data accessed with SQL, streaming, *etc.*

Apache Spark

Pros:

- ↗ **light-weight and fast**
- ↗ **industrial strength**
- ↗ **declarative functional style** for massive Hadoop/MapReduce computation
- ↗ **syntax-interfaced** with most popular scripting and analytics systems (Python, Java, Scala, R)
- ↗ is **gaining rapid popularity** in the Analytics tools landscape

Cons:

- ↘ still **relatively young**
- ↘ **no CP integration** (yet?)

And so. . .

Synthesis

Constraints meet Analytics:

GREAT IDEA!

What's the best way?

And so. . .

CP meets Analytics—Use Cases

IBM ILOG Solver + IPython Notebook Mixing CP/OR tools

(pros and cons)

Google OR Tools

(pros and cons)

Scripting CP and Analytics

- **Use case 1**
  IBM ILOG Optimization Decision Manager Enterprise
  Using IPython Notebook for analytics script with distributed multithreaded CP with IBM Solver
  (JF Puget, IBM)
  - *“IT Best Kept Secret Is Optimization”*
  - Solving Optimization Problems on the Cloud with Python
    (Apr 13, 2015)
  - A Sudoku Web App Based On DOcloud and Python
    (Apr 27, 2015)
- **Use case 2**
  Google OR tools
  OR models scripted with Python/Java/C/C++/C#
  (Laurent Perron, Google) – CP 2013

IBM ILOG Optimization Decision Manager Enterprise

Pros:

- ↗ **fastest existing CP/OR solvers** (ILOG/CPLEX)
- ↗ **industrial strength**
- ↗ uses IPython Notebook to **leverage Python for scripting CP with Analytics**

Cons:

- ↘ uses relatively **low-level tooling** for distributed concurrency management (Boot2Docker) (would prefer generic reusable higher-level declarative utilities for multithreaded concurrency)

Google OR Tools

Pros:

- ↗ **industrial strength** (load and time)
- ↗ scripting (Python, Scala, C#) makes up **4/5 of the code** for orchestrating C++-compiled solving modules **at 1/20 of the cost** of dedicated CP systems such as OPL or AMPL
- ↗ **full interfaces** with Python, Java/Scala (JVM), and C#

Cons:

- ↘ **no high-level** OR **model management** (Minizinc/Flatzinc to parse models and display existing solutions)
- ↘ **limited dynamicity** (relies on static presolving)
- ↘ **limited search control** adaptability (esp. local search)

And so. . .

Prognosis

CLP Declarative Scripting

Does it makes sense?

Leverage C(L)P: declarative scripting for CP/OR Analytics?

Two-way street:

1. Analytics extended with CP
2. CP extended with Analytics

- CP libraries for procedural languages
  - **Exemplar**: Python-CP libraries

- CLP scripting languages
  - **Exemplar**: Picat scripting

And so. . .

Declarative scripting for analytics

Where are we today?

(pros and cons)

Python-CP libraries

Pros:

- ↗ **no need for new syntax**—Python
- ↗ **light-weight**, dynamically typed and interpreted
- ↗ **flexible style** complete Python's already varied styles (procedural, functional, object-oriented) with CP style
- ↗ **full access to** Python **libraries**

Cons:

- ↘ **scripting itself is not declarative** nor generic (need to program an explicit solver + search interpreter per app)

Picat

- **P**attern-matching

  Predicates and functions are defined with pattern-matching rules

- **I**mperative

  Assignments, loops, list comprehensions

- **C**onstraints

  CP, SAT and LP/MIP

- **A**ctors

  Action rules, event Action rules, event-driven programming, actor driven programming, actor-based concurrency

- **T**abling

  Memoization, dynamic programming, planning, model-checking

Picat scripting

Picat script for traversing a directory tree:

```
import os.

traverse(Dir), directory(Dir) =>
    List = listdir(Dir),
    printf("Inside %s%n",Dir),
    foreach(File in List)
        printf(" %s%n",File)
    end,
    foreach(File in List, File != ".", File != "..")
        FullName = Dir ++ [separator()] ++ File,
        traverse(FullName)
    end.

traverse(_) => true.
```

Picat

Picat Sudoku solver—courtesy of Hakan Kjellerstrand

```
sudoku(N, Board) =>
    N2 = N*N,
    Vars = Board.vars(),
    Vars ::  1..N2,
    foreach(Row in Board)
       all_different(Row)
    end,
    foreach(Column in transpose(Board))
       all_different(Column)
    end,
    foreach(I in 1..N..N2, J in 1..N..N2)
       all_different([Board[I+K,J+L] : K in 0..N-1, L in 0..N-1])
    end,
    solve([ffd,down], Vars).   % ffd+down fastest var ordering
```

Picat

Pros:

↗ **light-weight**, dynamically typed and interpreted

↗ **flexible style** as appropriate: procedural (*if-then-else, looping, destructive assignment*) ***as well as*** functional, and constraint logic programming

↗ **terse**, clear, and easily reusable CP

↗ **most script-like** among CLP languages (unique in enhancing CP with built-in procedural scripting)

Cons:

↘ still **young**

↘ **unfamiliar syntax** (to majority of users)

↘ **needs** more **tools** (libraries)

↘ **needs** more **interfaces** (IPicat Notebook?)

And so. . .

Discussion

Declarative Scripting, CP/OR, Analytics?

Discussion

Mimimum Requirements

- ▶ **Industrial strength** (cloud-aware, large-scale concurrency)
- ▶ **Ease of use** (portability, graphing, reporting)
- ▶ **Immediate payoff** (notation, flat learning curve, instant compatibility with familiar tools)
- ▶ **Libraries** (provide a large varied pool of app-specific reusable tools)
- ▶ **Interfaces** (open up to the rest of the world)
- ▶ **Ontologies** (encode knowledge for models, solvers, search)

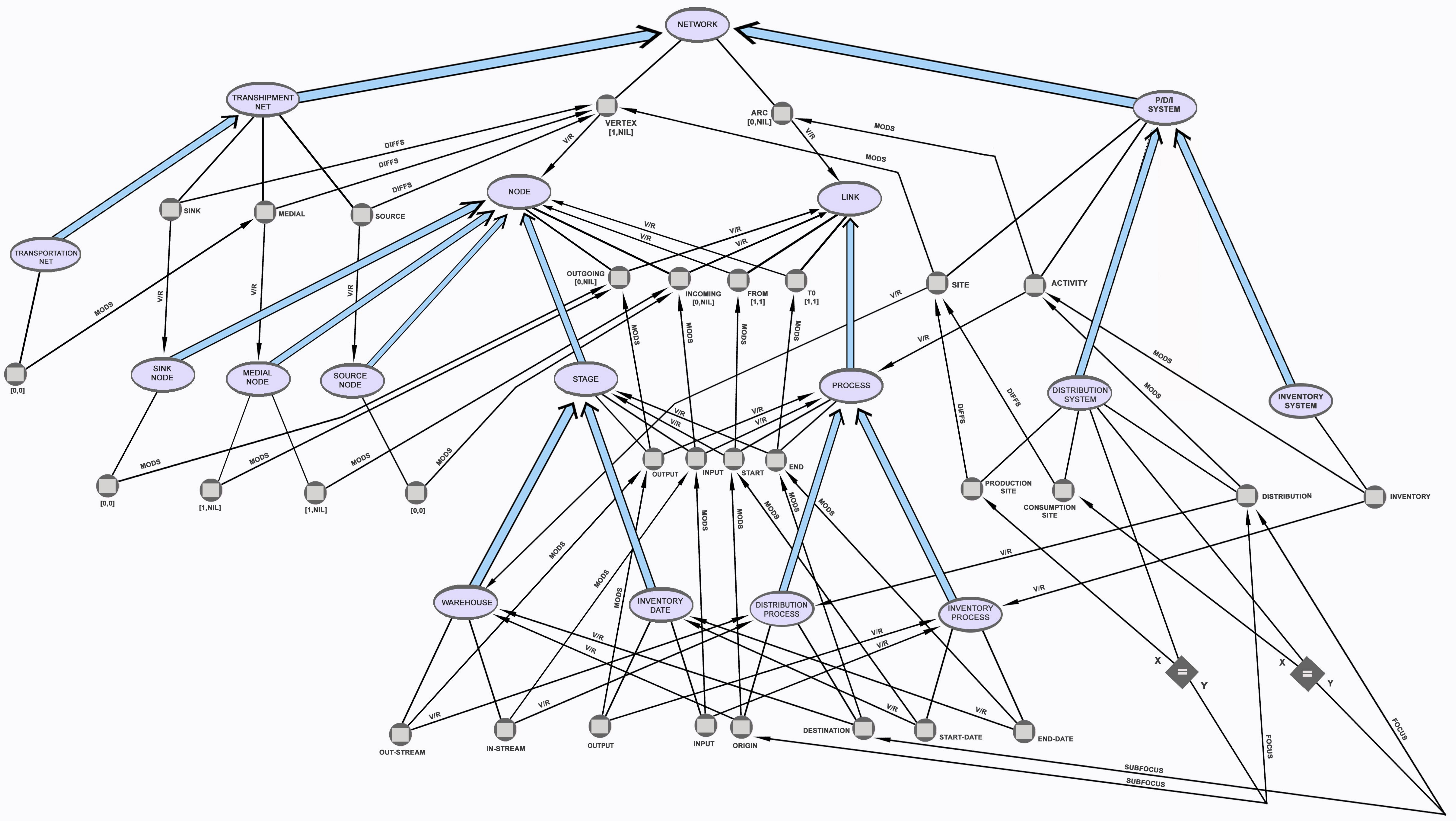
Discussion

Challenges for CP-based **declarative analytics**:

- need standard interfaces to easy plug-in modules for smooth **syntax-independent constructs for solving + searching** (heuristics libraries) with standard interfaces in most popular languages

- need **analytics scenario libraries** for reusable configurations (statistical *cum* CP/OR)

- need **ontologies for models, use scenarios, and search** to enable **knowledge-based model-building** (*e.g.*, PDI-net example)


NETWORK
TRANSHIPMENT NET
VERTEX [1,NIL]
ARC [0,NIL]
P/D/I SYSTEM
DIFFS
DIFFS
DIFFS
MODS
MODS
V/R
V/R
NODE
LINK
SINK
MEDIAL
SOURCE
TRANSPORTATION NET
MODS
V/R
V/R
V/R
V/R
V/R
V/R
V/R
OUTGOING [0,NIL]
INCOMING [0,NIL]
FROM [1,1]
T0 [1,1]
SITE
ACTIVITY
V/R
V/R
MODS
MODS
MODS
MODS
MODS
[0,0]
SINK NODE
MEDIAL NODE
SOURCE NODE
STAGE
PROCESS
DISTRIBUTION SYSTEM
INVENTORY SYSTEM
DIFFS
DIFFS
MODS
MODS
V/R
V/R
V/R
V/R
OUTPUT
INPUT
START
END
PRODUCTION SITE
CONSUMPTION SITE
DISTRIBUTION
INVENTORY
MODS
MODS
MODS
MODS
[0,0]
[1,NIL]
[1,NIL]
[0,0]
MODS
MODS
MODS
MODS
MODS
MODS
MODS
V/R
V/R
WAREHOUSE
INVENTORY DATE
DISTRIBUTION PROCESS
INVENTORY PROCESS
V/R
V/R
V/R
V/R
X
=
Y
X
=
Y
V/R
V/R
V/R
V/R
OUT-STREAM
IN-STREAM
OUTPUT
INPUT
ORIGIN
DESTINATION
START-DATE
END-DATE
FOCUS
FOCUS
SUBFOCUS
SUBFOCUS

Discussion

So what about **“declarative scripting for CP/Analytics?”**

- **CP/OR constraint-based analytics scripting has become essential** in actual field deployment for decision-making (it connects models with actual data, carries out statistical, “what-if,” and sensitivity analyses, produces reports, plots, justifications, *etc.* . . .)
- **C(L)P has also started to be applied to Analytics** as its style enhances expressive power (high-level, declarative); though still needs work to reach popularity

Discussion

But what about existing scripting languages?

Many, many, “scripting” languages are used in Analytics, but most essentially provide similar **homomorphic syntax** for:

- **data types**
  (monoid comprehensions, esp., collections, arrays, tables)

- **functional computation over collections**
  great for concurrency (MapReduce, multi-D array algebra)

- even **“procedural” iteration with assignment** can be cast as a monoid comprehension

Discussion

Rather than **many-to-many *ad hoc* interfaces**, it makes more sense to agree on **one** essential canonical (abstract) structure and operations (*e.g.*, comprehension syntax)

It is easier to have n (homomorphic) interfaces than n^2 *ad hoc* translators; and only one canonical representation to conform to than n *ad hoc* ones

Work such as Bistarelli/Rossi makes CP based on semi-rings (which BTW extend collection monoids) a “natural” canonical algebra for “soft” CP (including Fuzzy Sets, Bayesian, GDL, Rough Sets, etc., . . .) So one could argue that **CP has the means to make such “non-crisp” analysis possible** by setting the CP solving in the appropriate algebra(s).

Discussion—Recapitulation

- **Where we are**:
  - silo-ed CP systems are dead: too hard to interface with GP middleware, analytics, graphing, and reporting (*e.g.*, OPL, AMPL)
  - flexible Analytics combines CP/OR and Statistical Analysis via light-weight orchestrating scripts
  - *Ergo*: scripting is *the* key for orchestrating CP apps
- **What we need**:
  - **disciplined scripting** (not for just CP): simple, terse, and easy to (re)use
  - **knowledge-based scripting** for Analytics: ontologize collection algebras and statistics
  - **declarative scripting** for CP/Analytics: ontologize models, solving, and search

Discussion—Conclusion

This last item—**ontologizing CP/OR—is the most sensible way** IMHO; *i.e.*,

The Global Constraint Catalog as an *attributed ontology* *à la* FCA to be used operationally **for declarative scripting as *"OntoLogic"* Programming** (*e.g.*, CLP *à la* LIFE)

And BTW: ontological reasoning itself *is* CP!

Lest the cobbler's children stay the worst shod. . .

Thank You For Your Attention !