# Constraint Programming with Mozart – An Appetizer

KTH Microelectronics and Information Technology

**Christian Schulte**
KTH – Royal Institute of Technology, Sweden
`schulte@imit.kth.se`

## Goal of Appetizer

- Underlying principles
- Sketch of how to do in Oz
- Modeling techniques
- Mozart advantages and disadvantages

…no time for hands-on tutorial

## Constraint Programming

- Modeling and solving combinatorial problems

**start with a first toy problem**

## Send More Money (SMM)

- Find distinct digits for letters, such that

```
    SEND
+   MORE
= MONEY
```

## Constraint Model for SMM

- Variables and values
  `S,E,N,D,M,O,R,Y ∈ {0,…,9}`
- Constraints

```
distinct(S,E,N,D,M,O,R,Y)
            1000×S+100×E+10×N+D
 +          1000×M+100×O+10×R+E
 = 10000×M+1000×O+100×N+10×E+Y
S≠0       M≠0
```

## Solving SMM

- Find values for variables
  such that
  **all constraints satisfied**
- Enumerate values, test constraints…
  …poor: we can do better than that!

## Constraint Programming

- Compute with set of **possible** values
  - as opposed to assignments
- Prune impossible values
  - constraint propagation
- Search
  - distribute — search tree of simpler subproblems
  - explore — find solution in tree

## Propagation for SMM

- Results in

```
S=9   E∈{4,…,7}  N∈{5,…,8}  D∈{2,…,8}
M=1   O=0        R∈{2,…,8}  Y∈{2,…,8}
```

- Propagation **alone** not sufficient!
  - create simpler sub-problems
  - distribution and exploration

## Overview

- Principles
  - constraint propagation
  - search
- Summary of principles and significance
- Modeling techniques
- Oz and Mozart

## Principles: Constraint Propagation

## Important Concepts

- Constraint store
- Basic constraint
- Propagator
- Non-basic constraint
- Constraint propagation

## Constraint Store

x∈{3,4,5}  y∈{3,4,5}

- Stores basic constraints

  map variables to possible values

## Constraint Store

finite domain constraints

x∈{3,4,5} y∈{3,4,5}

- Stores basic constraints
  - map variables to possible values

## Constraint Store

x∈{3,4,5} y∈{3,4,5}

- Stores basic constraints
  - map variables to possible values
- Domains: finite sets, real intervals, trees, …

## Propagators

- Implement non-basic constraints

  `distinct(x`$_1$`,…,x`$_n$`)`

  `x + 2×y = z`

  - smart algorithmic components

## Propagators

x≥y　　y>3

x∈{3,4,5} y∈{3,4,5}

- Amplify store by constraint propagation

## Propagators

x≥y　　y>3

x∈{3,4,5} y∈{3,4,5}

- Amplify store by constraint propagation

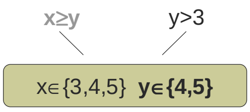
## Propagators

x≥y　　y>3

x∈{3,4,5} y∈{4,5}

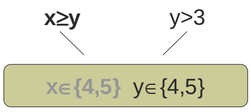
- Amplify store by constraint propagation

## Propagators



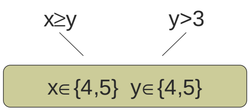

- Amplify store by constraint propagation

## Propagators



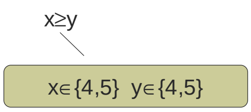

- Amplify store by constraint propagation

## Propagators



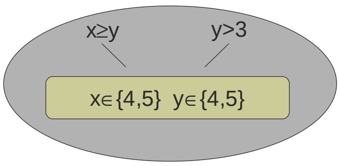

- Amplify store by constraint propagation
- Disappear when done (entailed)
  - no more propagation possible

## Propagators




- Amplify store by constraint propagation
- Disappear when done (entailed)
  - no more propagation possible

## Computation Space



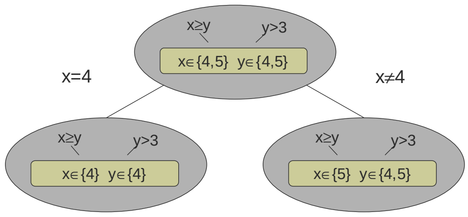

- Store with connected propagators

## Principles: Search

## Important Concepts

- Distribution
- Exploration
- Heuristics
- Best solution search

## Distribution (Branching)




- Yields spaces with additional constraints
- Enables further constraint propagation

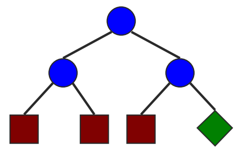
## Distribution Strategy

- Pick variable x with at least two values
- Pick value n from domain of x
- Distribute with

  x=n and x≠n

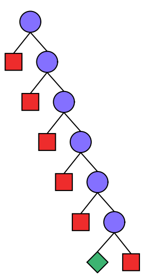
- Part of model

## Search



- Iterate propagation and distribution
- Orthogonal: distribution ⇆ exploration
- Nodes:
  - **Unsolved** • **Failed** • **Succeeded**

## SMM: Solution



```
    SEND
+   MORE
= MONEY

    9567
+   1085
= 10652
```

## Solving SMM in Oz

- Program script
  - script implements model
  - unary procedure: argument (root variable) is solution
- Script
  - introduce variables
  - basic constraints
  - post constraints
  - create branching

## Oz Script for SMM: Solution and Basic Constraints

```
proc {SMM Sol}
   S E N D M O R Y
in
   Sol=smm(s:S e:E n:N d:D m:M o:O R:r y:Y)
   Sol ::: 0#9
   …
end
```

## Oz Script for SMM: Post Propagators

```
proc {SMM Sol}
   …
   {FD.distinct Sol}
   S\=:0 M\=:0
              1000*S+100*E+10*N+D
            + 1000*M+100*O+10*R+E
   =: 10000*M+1000*O+100*N+10*E+Y
   …
end
```

## Oz Script for SMM: Distribution Strategy

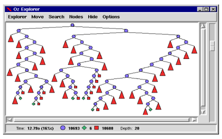
```
proc {SMM Sol}
   …
   {FD.distribute naive Sol}
end
```

## Complete Oz Script for SMM

```
proc {SMM Sol}
   S E N D M O R Y
in
   Sol=smm(s:S e:E n:N d:D m:M o:O R:r y:Y)
   Sol ::: 0#9
   {FD.distinct Sol}
   S\=:0 M\=:0
              1000*S+100*E+10*N+D
             +1000*M+100*O+10*R+E
   =: 10000*M+1000*O+100*N+10*E+Y
   {FD.distribute naive Sol}
end
```

## Solving SMM in Oz

```
{ExploreOne SMM}
```

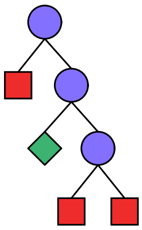


- Use Oz Explorer
  - interactive, visual search
  - allows access to nodes in search tree
  - gain insight into propagation and distribution
- Other engines available

## Heuristics for Distribution

- Which variable
  - least possible values (first-fail)
  - application dependent heuristic
- Which value
  - minimum, median, maximum
    x=m or x≠m
  - split with median m
    x<m or x≥m
- Problem specific

## SMM: Solution With First-fail



```
    SEND
+   MORE
= MONEY

    9567
+   1085
= 10652
```

## Send Most Money (SMM++)

- Find distinct digits for letters, such that

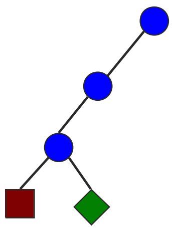
```
    SEND
+   MOST
= MONEY
```

and MONEY maximal

## Best Solution Search

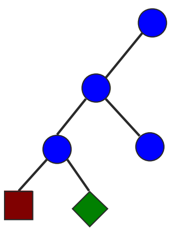
- Naïve approach:
  - compute all solutions
  - choose best
- Branch-and-bound approach:
  - compute first solution
  - add “betterness” constraint to open nodes
  - next solution will be “better”
  - prunes search space

## Branch-and-bound Search



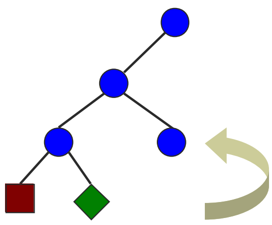
- Find first solution

## Branch-and-bound Search



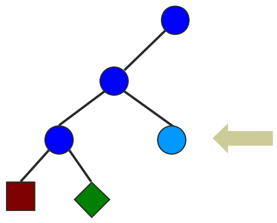
- Explore with additional constraint

## Branch-and-bound Search



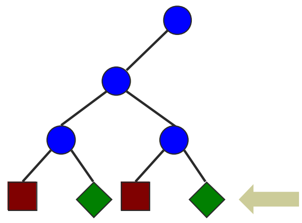
- Explore with additional constraint

## Branch-and-bound Search



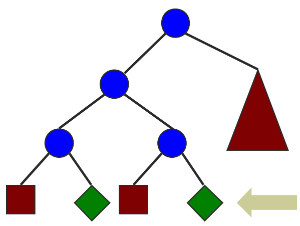
- Guarantees better solutions

## Branch-and-bound Search



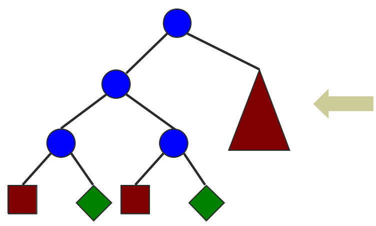
- Guarantees better solutions

## Branch-and-bound Search



- Last solution best

## Branch-and-bound Search



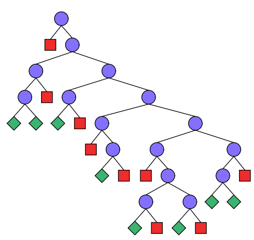
- Proof of optimality

## Modeling SMM++

- Constraints and branching as before
- Order among solutions with constraints
  - so-far-best solution `S,E,N,D,M,O,T,Y`
  - current node **`S,E,N,D,M,O,T,Y`**
  - constraint added

```
10000×M+1000×O+100×N+10×E+Y
    <
10000×M+1000×O+100×N+10×E+Y
```

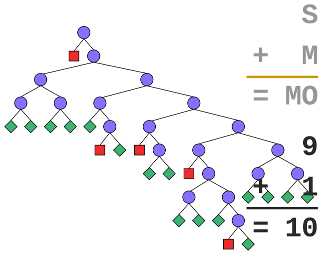
## SMM++: Branch-and-bound



```
    SEND
+   MOST
= MONEY

    9782
+   1094
= 10876
```

## SMM++: All Solution Search



```
   SEND
+  MOST
= MONEY

   9782
+  1094
= 10876
```

## Summary

## Modeling and Solving

applications

principles

- Modeling
  - variables with domain
  - constraints to state relations
  - branching strategy
  - solution ordering
- Solving
  - constraint propagation
  - constraint branching
  - search tree exploration

## Constraint Programming in Oz

- Script
  - implements constraint model
- Solution order
  - defined by binary procedure
- Exploration
  - interactive: Oz Explorer
  - search module: plain, best, parallel, …

## Application Areas

- Timetabling
- Scheduling
- Crew rostering
- Resource allocation
- Workflow planning and optimization
- Gate allocation at airports
- Sports-event scheduling
- Railroad: track allocation, train allocation, schedules
- Automatic composition of music
- Genome sequencing
- Frequency allocation
- …

## Why Does CP Matter?

- Middleware for combining smart algorithmic components (propagators)
  - scheduling
  - graphs
  - flows
  - …

  …for strong propagation
- Essential extra constraints…

  …for flexibility

## SMM: Strong Propagation

```
   SEND
+  MORE
= MONEY

   9567
+  1085
= 10652
```

## Significance

- Constraint programming identified as a strategic direction in computer science research
  - [ACM Computing Surveys, December 1996]
- Applications are ubiquitous

# Modeling

## Modeling Strategy

- Understand problem
  - identify variables
  - identify constraints
  - identify optimality criterion
- Attempt initial model    simple
  - try on examples to assess correctness
- Improve model    much harder
  - scale up to real problem size

## Modeling Techniques

- Find variables and values
  - decrease symmetries
  - dual models: change values and variables
  - combine models: channeling
- Increase propagation
  - strong methods
  - redundant (implied) constraints but non-redundant propagation

## Modeling Techniques

- Remove useless solutions
  - symmetrical: symmetry breaking
  - same cost: dominance constraints
- Good heuristic for distribution
  - which variable: size, degree, regret, …
  - how to split domains: single value, bisection, …
  - in which order to split: minimum, median, maximum, …

# Oz and Mozart

## Getting Started with Mozart

- Use tutorial shipped with Mozart
  - Schulte, Smolka. Finite Domain Constraint Programming in Oz. A Tutorial.
- Little knowledge on Oz required
  - scripts are unary procedures
  - orders are binary procedures
  - introducing variables
  - conditional statements
  - calling functions and procedures
  - tuples (records) for solutions
  - loops for iterating over tuples

## Mozart Features

- Finite domain integers
  - general purpose: arithmetic, …
  - scheduling
- Finite sets
- Search: orthogonal exploration
  - basic + interactive + parallel + …
- Tools
  - OPI, Explorer, Browser, Inspector, …

## Mozart Advantages

- Incremental and interactive development
  - understand problem and refine model
  - rich tool support
- Integration with concurrency and distribution
  - multi agent applications
- Well documented
- Freely available
- Programmable and Extensible

## Programmable and Extensible

- Programming [Oz]
  - scripts
  - distribution
  - exploration (Explorer, parallel search, …)
  - combination mechanisms
- Extending [CPI in C++]
  - propagators
  - variables

## Mozart Disadvantages

- Small set of good propagators
  - "global constraints"
  - will worsen due to lack of contributors
- Inflexible interface for propagators
  - unrealistic assumptions
- Initial burden to learn Oz
- Not easy to embed

# Summary

## Constraint Programming with Mozart

- Powerful technology for combinatorial optimization
- Mozart free, programmable, and accessible system for constraint programming
  - requires more propagators
- Most effort is in modeling (understanding)
  - not dependent on Oz and Mozart