

Constraint Programming

Aufgaben Beschreiben statt selbst Lösungen programmieren

Heiko Spindler (IT-Berater)



Sudoku
Maschinenbelegung
Systemkonfiguration
Routenplanung
Teambesetzung
Verifikation
Aufgabenverteilung
Ressourcenallokation
Frequenzzuweisung
Mitarbeiterplanung
Testfallgenerierung
Bewegungsplanung
Zeitfensterprobleme
Validierung
Netzwerktopologie
Tourenplanung
Netzwerkconfiguration
Steuerung
Schichtplanung
Brettspiele
Raumplanung
Projektzeitplanung
Transportoptimierung
Netzwerke
Produktconfiguration
Kreuzworträtsel

Agenda

- Einführung in Constraint Programming
- Variablen und Constraints
- Ein erstes Beispiel
- Frameworks
- Beispiel: Rätsel lösen
- Abstrakte Sprachen für CP
- Beispiel: Scheduling / Timetables
- Beispiel: Prozess-Ablauf-Planung
- Fazit

Constraint Programming (CP)

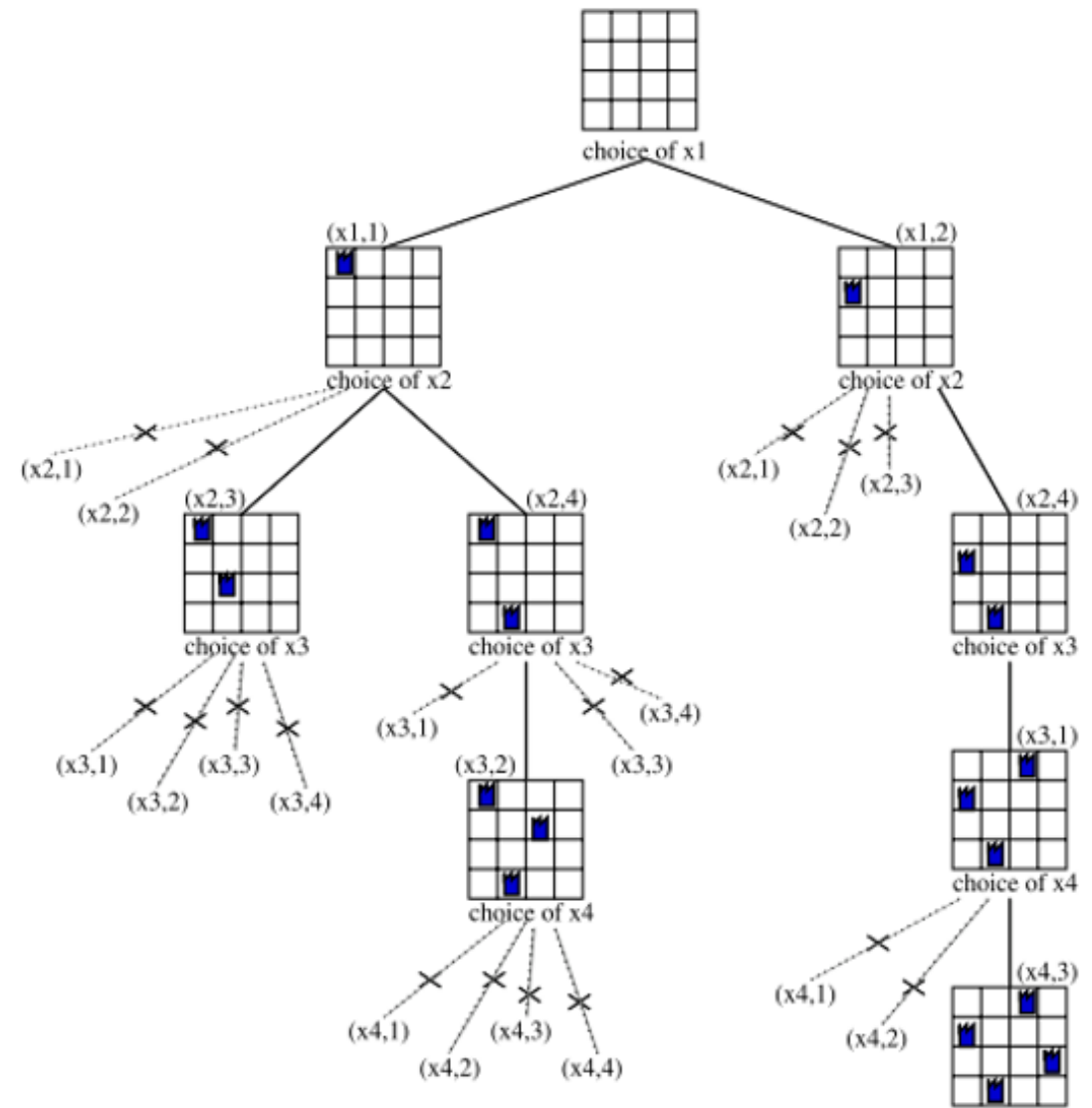
- Ein Constraint Satisfaction Problem (CSP) ist eine Tripple $\langle X, D, C \rangle$, für das gilt:
 - $X = \{X_1, \dots, X_n\}$ ist eine Menge von Variablen.
 - D ist eine Funktion, die jeder Variablen einen Wertebereich (Domain) zuweist.
 - C ist eine Menge von Bedingungen (Constraints).
- Ein CSP kann leicht in ein Constraint Optimization Problem (COP) umgewandelt werden:
 - Bewertung der gefundenen Lösungen vornehmen



Bedingungen (Constraints)

- Gesucht werden für alle Variablen konkrete Werte aus dem jeweiligen Wertebereich (Domain) für die alle Bedingungen (Constraints) erfüllt sind.
 - Eine Bedingung (Constraint) definiert eine Beziehung zwischen Variablen.
-

Wie arbeiten CP Solver?



Beispiel Mathematische Formeln lösen

- Variablen: a, b, c
- Domäne: a, b, c sind Integer aus dem Wertebereich (z.B. 1..100)
- Constraint: $3ab+7bc+9ca = 91$

Coding: Arithmetische Constraints

```
Model model = new Model("Formula Solver");
```

```
// Define integer variables for a, b, and c
```

```
IntVar a = model.intVar("a", 1, 10); // Set bounds for demonstration
```

```
IntVar b = model.intVar("b", 1, 10);
```

```
IntVar c = model.intVar("c", 1, 100);
```

```
// 3ab+7bc+9ca = 91
```

```
a.mul(b).mul(3).add( b.mul(c).mul(7)).add(c.mul(a).mul(9)).eq(91).post();
```

```
// Print the solution if found
```

```
if (model.getSolver().solve()) {  
    System.out.println("Solution found:");  
    System.out.println("a: " + a.getValue());  
    ...  
}
```

Arten von Constraints

- **Arithmetische Constraints:**
 - definieren Beziehungen zwischen Variablen mit arithmetischen Operationen und Relationen (z. B. Gleichheit, Ungleichheit, kleiner als, größer als).
- **Logische Constraints:**
 - formulieren logische Beziehungen zwischen Variablen (z. B. UND, ODER, NICHT).
- **Globale Constraints:**
 - drücken komplexe Beziehungen zwischen mehreren Variablen aus, wie z. B. „allDifferent“, „cumulative“.
- **Tabellen Constraints:**
 - definieren erlaubte Wertekombinationen für einen Satz von Variablen.
- **Reguläre Ausdrücke**
- **Graphen**

Choco-solver

- Kostenlose Open-Source-Java-Bibliothek für Constraint-Programmierung.
- Der Benutzer modelliert sein Problem deklarativ, indem er die Constraints definiert, die in jeder Lösung erfüllt werden müssen.
- Anschließend wird das Problem durch abwechselnde Constraint-Filteralgorithmen und einen Suchmechanismus gelöst.
- <https://choco-solver.org/>



Choco-Solver mit Python API

- <https://pypi.org/project/pychoco/>
- The pychoco library uses a *native-build* of the original Java Choco-solver library, in the form of a shared library, which means that it can be used without any JVM. This native-build is created with [GraalVM](#) native-image tool.

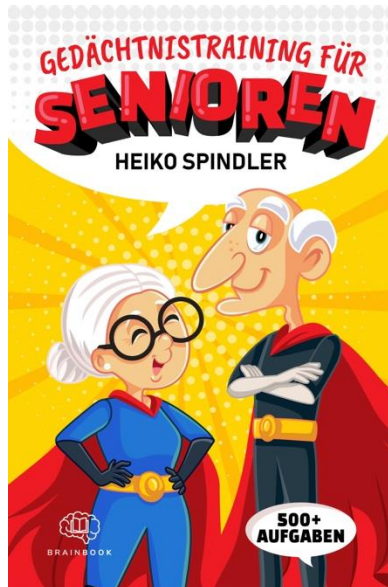


Alternative Frameworks

- JaCoP
 - „Java Constraint Programming solver“
 - <https://github.com/radsz/jacop>
 - Letzte Änderungen im Jahr 2023
- ILOG Solver (www.ilog.fr)
- GECODE (www.gecode.org)
- <https://developers.google.com/optimization>
- <https://github.com/chuffed/chuffed>
- Rust: <https://github.com/ptal/pcp>
- C++: <https://github.com/pothitos/naxos>

Und viele mehr ...

Die HirnSport.de - Bücher



Choco-Solver: Sokudo-Beispiel

		8				7		
	3	9				4	6	
7	5		9		4		3	8
		5		4		2		
			3		5			
		1		6		8		
4	6		8		1		9	7
	1	3				6	8	
		7				5		

Coding: Implementierung Sudoku

```
Model model = new Model("sudoku");
```

```
...
```

```
for (int row = 0; row != SIZE; row++) {  
    for (int col = 0; col != SIZE; col++) {  
        int value = predefinedRows[row][col];  
        // is this an unknown? if so then create it as a bounded variable  
        if (value < MIN_VALUE) {  
            grid[row][col] = model.intVar(format("[%s.%s]", row, col), MIN_VALUE,  
MAX_VALUE);  
        } else {  
            // otherwise we have an actual value, so create it as a constant  
            grid[row][col] = model.intVar(value);  
        }  
    }  
}
```

Coding: Implementierung Sudoku

```
for (int i = 0; i != SIZE; i++) {  
    model.allDifferent(getCellsInRow(grid, i)).post();  
    model.allDifferent(getCellsInColumn(grid, i)).post();  
    model.allDifferent(getCellsInSquare(grid, i)).post();  
}
```

...

```
Solver solver = model.getSolver();  
solver.solve();
```

...

Latin-Square mit Summen

Die hellen Zellen enthalten die Zahlen von 1 bis 9.

Die grauen Zellen enthalten die Summe der Zeile, Spalte bzw. Diagonale.

			15
1			12
3		7	18
8	23	14	17

Coding: Implementierung Latin-Square mit Summen

```
for (int row = 0; row < SIZE; row++) {  
    for (int col = 0; col < SIZE; col++) {  
        if (exercise.gridEx[row][col] > 0) {  
            grid[row][col] = model.intVar(exercise.grid[row][col]);  
        } else {  
            grid[row][col] = model.intVar(format("[%s.%s]", row, col), MIN_VALUE, MAX_VALUE);  
        }  
    }  
}
```

```
model.allDifferent(getAllCells(grid)).post();  
for (int rowColumnIndex = 0; rowColumnIndex < SIZE; rowColumnIndex++) {  
    model.sum(getCellsInRow(grid, rowColumnIndex), "=",  
exercise.sumRows[rowColumnIndex]).post();  
    model.sum(getCellsInColumn(grid, rowColumnIndex), "=",  
exercise.sumColumns[rowColumnIndex]).post();  
}  
model.sum(getDiagonalsCells(grid), "=", exercise.sumDiagonal).post();
```

Analysieren der Lösungssuche

// Einbau eines Observer am Model

```
model.getSolver().setEventObserver( new  
LoggingEventObserver() );
```

Ausgaben des Observer dokumentieren das Vorgehen der Lösungssuche:

Removing value 12 from [0.0] = {1..16}

Updating lower bound to 15 from 2 for [1.1]

...

Begrenzen der Suche

- `limitTime`
 - Begrenzen nach vorgegebener Zeit
- `limitSolution`
 - Begrenzen nach Anzahl gefundener Lösungen
- Spezielle Begrenzungen
 - `limitNode`
 - `limitFail`
 - `limitBacktrack`

Abstrakte Sprachen für das Formulieren von Constraints

- MiniZinc is a free and open-source constraint modeling language
 - <https://github.com/minizinc>
 - <https://www.minizinc.org/index.html>
 - MiniZinc IDE: <https://play.minizinc.dev/>
- FlatZink
- XCSP (<https://xcsp.org/>)
- ...

MiniZinc IDE Demo

Run 

Compile

Latest 

Solver: Gecode 6.3.0  

Playground.mzn  

1 % Use this editor as a MiniZinc scratch book

2 int: n; % Anzahl der Gegenstände

3

4 array[1..n] of int: weight; % Gewicht der Gegenstände

5 array[1..n] of int: volume; % Volumen der Gegenstände

6 array[1..n] of int: value; % (Optional) Wert der Gegenstände

7

8 int: max_weight; % Maximales Gewicht des Rucksacks

9 int: max_volume; % Maximales Volumen des Rucksacks

10

11 array[1..n] of var 0..1: take; % 1 wenn Gegenstand i eingepackt wird

12

13 constraint

14 sum(i in 1..n)(weight[i] * take[i]) <= max_weight /\

15 sum(i in 1..n)(volume[i] * take[i]) <= max_volume;

16

17 n = 5;

Hide all

dzn

▼ Running Playground.mzn

take = [0, 0, 0, 0, 0];

take = [1, 0, 0, 0, 0];

take = [1, 1, 0, 0, 0];

take = [1, 0, 1, 0, 0];

=====

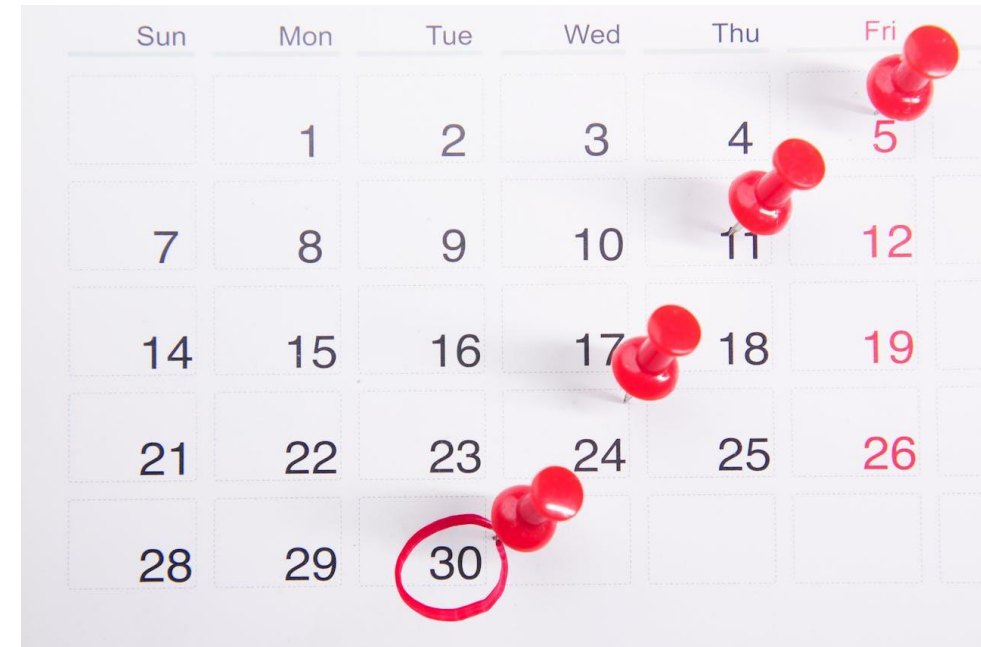
Finished in 191msec.

Anwendungsbeispiel: Timetable / Scheduling

Anforderungen

9 Personen teilen sich den Einsatz pro Monat auf:

- genau 2 Personen am Samstag verfügbar
- genau 1 Person am Montag - Freitag und Sonntag
- min. 2 Tage frei nach einem Einsatz
- jede Person soll zwischen 3 und 5 Einsätze pro Monat haben
- jede Person soll mindestens 1 Einsatz am Samstag haben
- über einen "Score" werden die Einsätze vergütet:
 - 2 Punkte: Montag-Donnerstag
 - 3 Punkte: Freitag und Sonntag
 - 4 Punkte: Samstag



Anwendungsbeispiel: Timetable / Scheduling

Days:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	31	Score	Duties
P: 0			3			2			2									11	4					
P: 1					2						2					2			12	5				
P: 2												4					3	11	4					
P: 3		4					3				2			3					12	4				
P: 4								3							2				11	4				
P: 5							4					3		2					11	4				
P: 6	3				2								4						12	4				
P: 7			2						2										11	4				
P: 8		4						4											11	3				

Anwendungsbeispiel: Optimierung einer hypothetischen Prozess-Ablauf-Planung



- Prozesse bestehen aus sequenziellen Schritten.
- Jeder Schritt hat eine Dauer und einen Bedarf an Ressourcen.
- Je Zeiteinheit darf die max. Kapazität nicht überschritten werden (z.B. Energiebedarf, Bandbreite oder Speicherplatz).

Globales Ziel:

Alle anstehenden Prozesse so früh wie möglich vollständig abarbeiten.

Darstellung des Ablaufs von Schritten im Prozess

[A1]	[A1]	[A2]	[A2]	[A2]	[A3]	[A3]	[A3]	[A4]	[A4]	[A4]	[A5]	[A5]	[A5]	[__]	[__]
[A1]	[A1]	[A2]	[A2]	[A2]	[A3]	[A3]	[A3]	[A4]	[A4]	[A4]	[A5]	[A5]	[A5]	[__]	[__]
[A1]	[A1]	[A2]	[A2]	[A2]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]
[B1]	[B1]	[B2]	[B2]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]
[B1]	[B1]	[B2]	[B2]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]
[__]	[__]	[B2]	[B2]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]	[__]

- Spalten sind Zeiteinheiten
- Zeilen stehen für Bedarf an Ressourcen für eine Aufgabe

Zwei Ansätze bei der Planung

Einfache Optimierung

- Prozesse und ihre Schritte werden sequenziell in den Arbeitsablauf eingefügt.
 - Sucht eine passende Lücke für jeden neuen Task: So früh wie möglich.
- ➔ Lokale Optimierung mit wenig Aufwand

Globale Optimierung mit CP

- Alle Prozesse mit ihren Schritten sind bekannt für den Planungshorizont
 - Volle Freiheit bei der Einplanung
- ➔ Globale Optimierung

Vergleich der beiden Verfahren

[illegible]

MaxTime: 1538

Optimale Lösung gefunden (minimales Ende aller Prozesse):

Gesamte Endzeit: 1406

[Z1][Z1][Z1][Z1][d2][g1][f1][f1][f1][f1][Z2][Z2][Z2][Z2][Z3][Z3][Z3][T1][T1][g4][g5][Z4][Z4][Z5][Z5][Z5][Z5][Z6][Z6][J1][J1][J1][Z7][Z7][Z7][Z7][Z1][Z1][Z1][Z1][d2][g1][f1][f1][f1][f1][Z2][Z2][Z2][Z2][Z3][Z3][Z3][T1][T1][g4][g5][Z4][Z4][Z5][Z5][Z5][Z5][Z6][Z6][J1][J1][J1][Z7][Z7][Z7][Z7][Z1][Z1][Z1][Z1][d2][g1][g1][g1][g1][g2][Z2][Z2][Z2][Z2][g3][g3][g4][T1][T1][g4][g5][Z4][Z4][Z5][Z5][Z5][Z5][Z6][Z6][g7][g7][g7][Z7][Z7][Z7][Z7][G1][G1][d1][d2][d2][g1][g1][g1][g1][g2][Z2][Z2][Z2][Z2][g3][g3][g4][T1][T1][Z1][g5][Z4][Z4][Z5][Z5][Z5][Z5][Z6][Z6][g7][g7][g7][g8][g8][g9][g9][G1][G1][d1][d2][g2][d3][g1][g1][g1][g2][g2][g2][g2][g3][H1][H1][g4][g4][g4][Z1][Z1][g5][g5][g5][g6][g6][g6][g6][g7][g7][g7][g7][g8][g8][g9][g9][d1][d1][g1][d2][g2][d3][g1][g1][g1][g2][g2][g2][g2][g3][H1][H1][d5][g4][g4][Z1][Z1][g5][g5][g5][g6][g6][g6][g6][g7][g7][g7][g7][g8][g8][h1][h1][d1][d1][g1][d2][g2][g2][r1][r1][r1][d4][g2][g2][g2][d5][d5][d5][d5][g4][g4][d6][Z1][g5][g5][g5][g6][g6][g6][g6][g7][d9][d9][d9][d9][d10][h1][h1][g1][g1][g1][g1][g2][g2][r1][r1][r1][d4][g2][g2][g2][d5][d5][d5][d5][d6][d6][d6][d6][g5][g5][g5][g6][g6][g6][g6][g7][d9][d9][d9][d9][d10][d10][d1][g1][g1][___][g1][___][g2][d3][d3][d3][___][d4][d4][d4][d5][d5][d5][d5][d6][d6][d6][d6][d7][d7][d7][d7][d8][d8][d8][d8][d9][d9][d9][d9][d10][d10][d1][g1][g1][___][g1][___][g2][d3][d3][d3][___][d4][d4][d4][d5][d5][d5][___][d6][d6][___][d6][d7][d7][d7][d7][d8][d8][d8][d8][d9][d9][d9][d9][___][d10][d10]

Bewertung

- Globale Optimierung mit CP nutzt die vorhandene Kapazität besser aus.
- Gewinn bei der Auslastung der Kapazität ca. 8 – 12%

Integration von CP in Applikation

In den Anforderungen ist oft formuliert:

- Planung und Steuerung brauchen Zugriff auf **alle** Daten: Stammdaten (inkl. des aktuellen Zustands), Aufträge, ...

Erster Impuls:

- Steuerungskomponente mit direktem Zugriff auf die Daten ausstatten.

Bei genauer Analyse:

- Es reicht meist eine reduzierte Sicht auf die Daten.
-
- ➔ Erstellen einer speziellen Sicht auf die „notwendigen“ Informationen.
 - ➔ Damit kann die Steuerungskomponente leichter separiert werden.
 - ➔ Erleichtert die Simulation der Planung und Steuerung.

Fazit CP

- Bietet mächtige Möglichkeiten Aufgabenstellung formal und (möglichst) technik-neutral zu Beschreiben.
- Gute Möglichkeiten der Integration in Applikationen.
- Es gibt ausgereifte und eingeführte Frameworks.
- Bietet Möglichkeiten den Prozess der Lösungsfindung zu konfigurieren.
- Folien und Beispiele finden sich unter:
<https://github.com/brainbrix/cpdemo>