Cryptoarithmetisches Puzzle

GWV WiSe 2016/2017

Finn-Lasse Jörgensen, Frederik Wille, Tronje Krabbe

UHH

Intro

Intro

```
  THREE
+ THREE
+  FIVE
=ELEVEN
```

Probleme

- AC3 Implementieren
- Gute Constraints & Variablen finden

AC3-Implementation

AC3-Implementation

```
class Network(object):

    def __init__(self, variables):
        self.variables = variables
        self._arcs = None
        self.constraints = []
        self.unary_constraints = []
```

AC3-Implementation

```
class Variable(object):

    def __init__(self, domain=None, meta={}):
        if domain is None:
            self.domain = set()
        else:
            self.domain = domain
        self.meta = meta
        self.constraints = []
        self.arcs = set()
```

AC3-Implementation

```
class Constraint(object):

    def __init__(self, variables, cfunc):
        self.variables = variables
        self.cfunc = cfunc

    def is_satisfied(self, values):
        assert type(values) is list
        return self.cfunc(*values)
```

AC3-Implementation

```
class UnaryConstraint(object):

    def __init__(self, variable, cfunc):
        self.variable = variable
        self.cfunc = cfunc

    def is_satisfied(self, value):
        return self.cfunc(value)
```

AC3-Implementation

```
class Arc(object):

    def __init__(self, variable, constraint):
        self.variable = variable
        self.constraint = constraint
```

AC3-Implementation

```
class Arc(object):
    # ...

    def make_consistent(self):
        non_consistent = []
        other_variable = self.other_variable()

        # find non-consistent elements
        for elem in self.variable.domain:
            for other in other_variable.domain:
                if self.constraint.is_satisfied([elem, other]):
                    break
            else:
                # for's else is executed if the loop
                # was *not* interrupted by a break!
                non_consistent.append(elem)

        # remove non-consistent elements from domain
        for elem in non_consistent:
            self.variable.domain.remove(elem)
```

AC3-Implementation

```
class Network(object):
    # ...
    def gac(self):
        todo_arcs = self.arcs

        self.satisfy_unary_constraints()

        while len(todo_arcs) > 0:
            arc = todo_arcs.pop()

            if not arc.is_consistent():
                arc.make_consistent()

                for variable in self.variables:
                    for narc in variable.arcs:
                        if (narc.other_variable() == arc.variable
                           and narc.constraint != arc.constraint):
                            todo_arcs.add(narc)
```

Constraints & Variablen

Variablen

- Pro Buchstabe: eine Variable
- Pro Spalte jeweils: eine Überlauf-Variable & eine Summen-Variable

```
  THREE
+ THREE
+  FIVE
=ELEVEN
```

Variablen: $E, F, H, I, L, N, R, T, V$ plus je sechs Überlauf- und Summen-Variablen

Constraints

- Jede Buchstaben-Variable, deren Buchstabe irgendwo an erster Stelle steht: Ein `UnaryConstraint`, dass ihr Wert nicht 0 sein darf.
- Jede Buchstaben-Variable mit jeder anderen Buchstaben-Variable: Ein `Constraint`, dass die Werte ungleich sein müssen.
- Jede Summen-Variable: muss gleich der Variable unter dem Strich sein ↯

Questions?

References I

`http://artint.info/`
Poole, Mackworth