Constraint Programming: Constraint Store

Eric MONFROY

IRIN, Université de Nantes

Objectives

- to give an intuitive notion of store
- adequacy between local consistency and store representation
- justify name of local consistencies

Constraint store

Constraint store

- to store constraints in memory
- modifications are necessary (add constraint)
- for efficiency : is generally adapted to the local consistency
- must relate constraint to variables and/or vice-versa

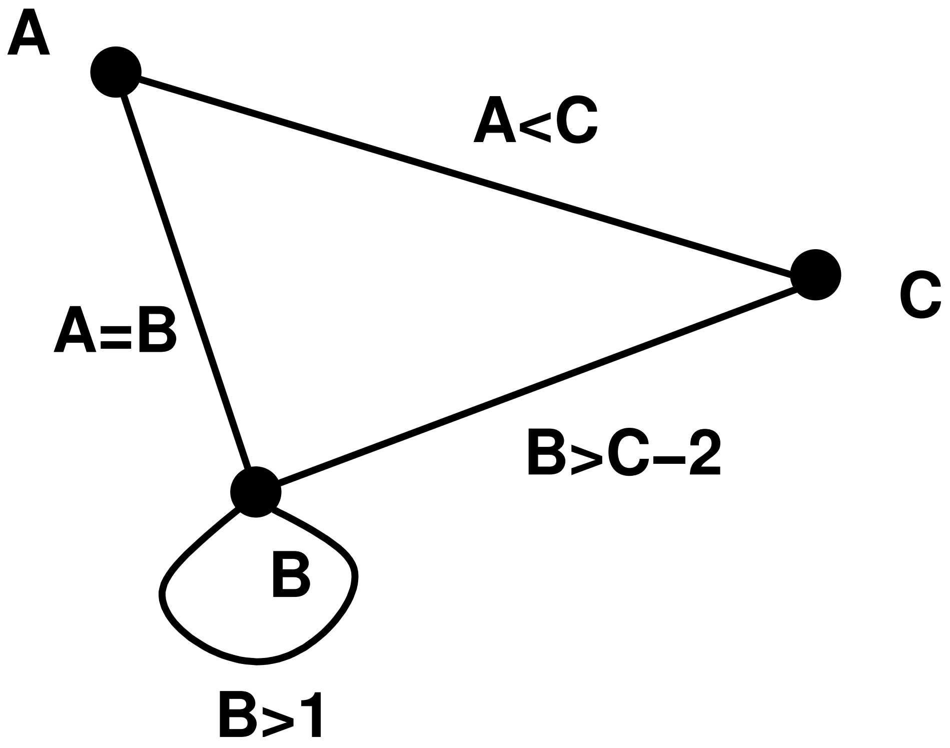
Store for binary constraint

- for consistency such as arc consistency
- from one variable, find related constraints
- from one constraint, find the 2 related variables

→ for arc consistency : when a variable is modified, find the variables that can be modified, i.e., the one link by an **arc**

Store for binary constraint

$$\{A = B, B > 1, B > C - 2, A < C; \quad \ldots\}$$



Store for binary constraint

- for consistency such as hyper-arc consistency
- from one variable, find related constraints
- from one constraint, find the n related variables

→ for hyper-arc consistency : when a variable is modified, find the variables that can be modified, i.e., the one link by an **hyper-arc**

Store for n-ary constraints

for n-ary constraints, with $n > 2$: use of the dual graph

transformation of :

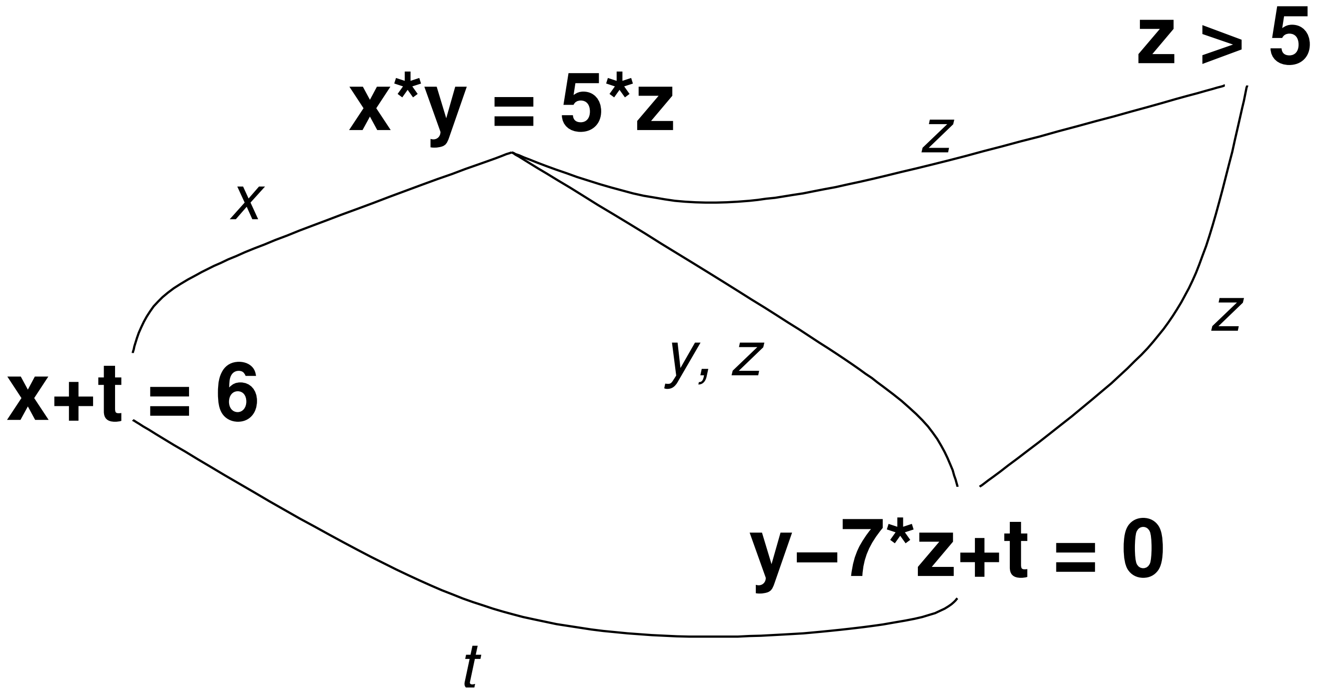
- node = variable
- arc = constraint

into

- node = contrainte
- arc = sharing of variables

Store for n-ary constraints

$$\{x + t = 6, x * y = 5 * z, y - 7 * z + t = 0, z > 5; \ \ \ldots\}$$



arcs from a constraint C : neighbourhood to re-invoke after *REVISE*(c) has modified a domain