Finite Set Constraints

Marco Kuhlmann & Guido Tack

Lecture 9

Plan for today

- finite set variables
- propagators for constraints on finite sets
- encoding binary relations
- encoding finite trees

Finite-set constraints

Finite-set variables

- A finite-domain integer variable
  takes values from a finite set of integers.
- A finite-domain set variable
  takes values from the power set of a finite set of integers.

Basic constraints

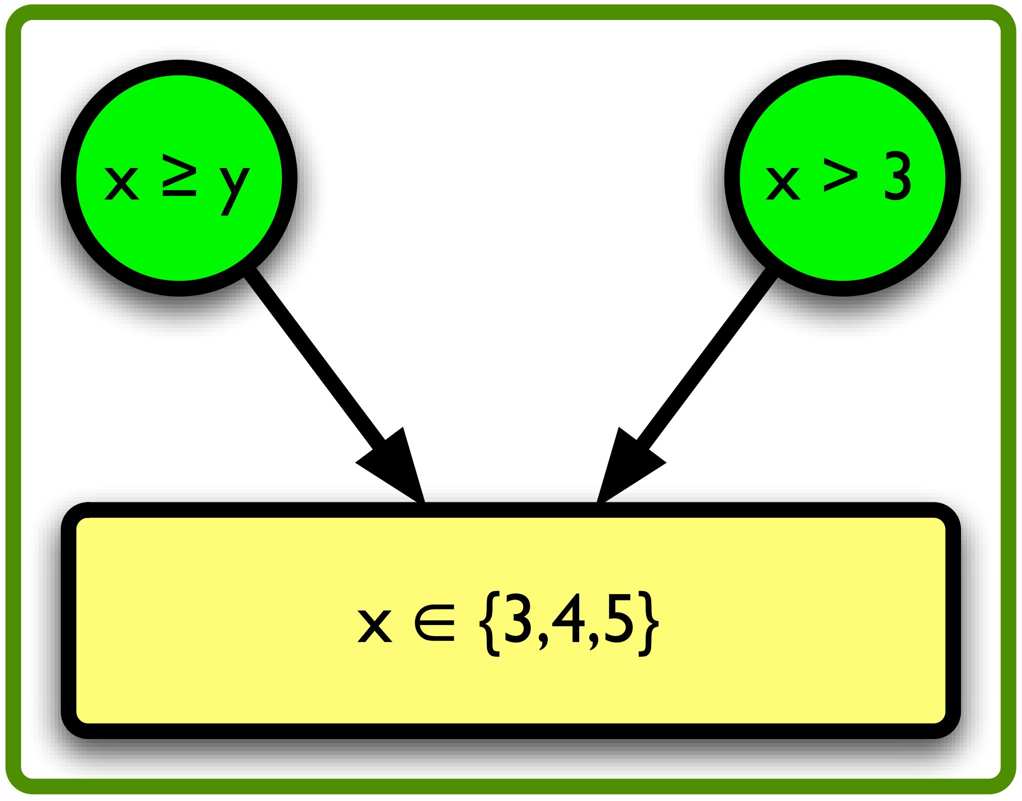
- approximation of assignments to FD variables:

$$I \in A$$

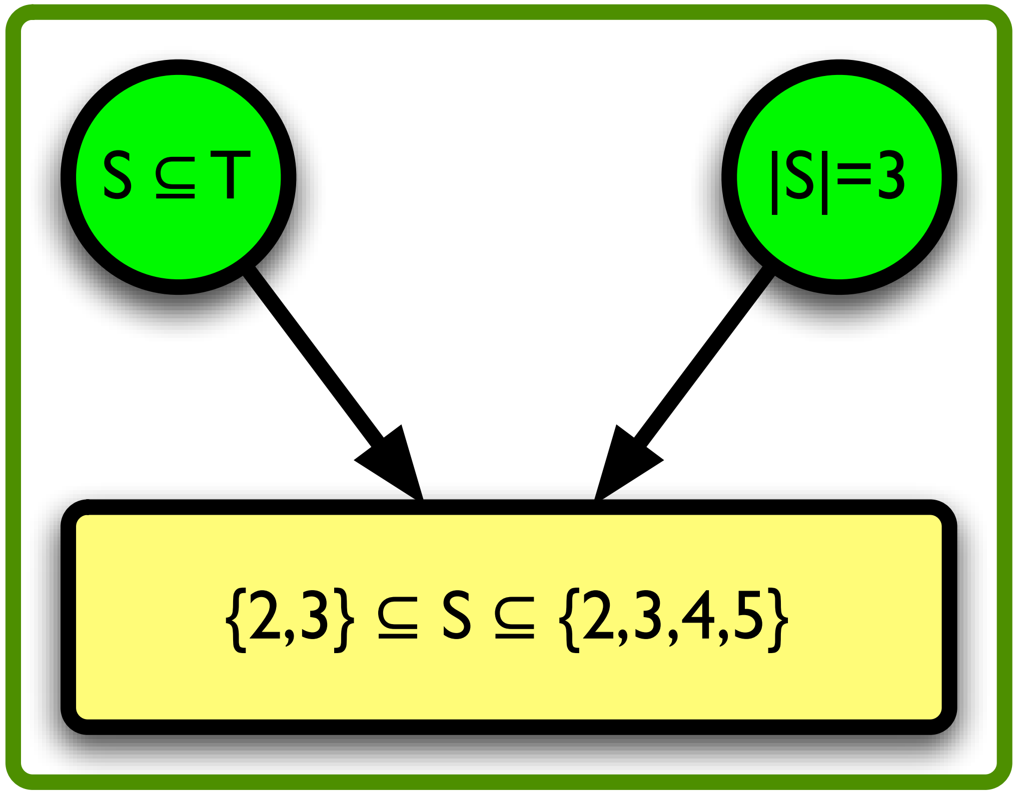
- approximation of assignments to FS variables:

$$A \subseteq S \qquad S \subseteq A$$

FD constraint store



FS constraint store



Approximate domains

- Let D be a domain.
- An **approximate domain** over D
  is a collection of subsets of D
  that contains D and is closed under intersection.
- For constraint propagation,
  we consider approximate domains that
  contain the empty set and all singletons.

Convex sets

- A **convex set** is a set of subsets of a domain D of integers that can be described by a greatest **lower bound** and a least **upper bound**:

$$C = \{ S \subseteq D \mid \lfloor C \rfloor \subseteq S \subseteq \lceil C \rceil \}$$

- The set of all convex sets of a domain D of integers forms an approximate domain over D.

Non-basic constraints

- express non-basic constraints in terms of basic constraints, using sets of inference rules
- express inferences in terms of the currently entailed lower and upper bounds

Subset constraint

$$S_1 \subseteq S_2$$

$$\frac{}{\lfloor S_1 \rfloor \subseteq S_2} \qquad \frac{}{S_1 \subseteq \lceil S_2 \rceil}$$

Subset constraint

$$S_1 \subseteq S_2$$

$$\frac{}{\lfloor S_1 \rfloor \subseteq S_2} \qquad \frac{}{S_1 \subseteq \lceil S_2 \rceil}$$

basic constraint

Specification of FS constraints

- intensional specification of constraints using formulae
- uses fragment of existential monadic second-order logic

$$
\begin{aligned}
Q_2 &::= \exists S \,.\, Q_2 \mid Q_1 \\
Q_1 &::= \forall x \,.\, B \mid Q_1 \wedge Q_1 \\
B &::= B \wedge B \mid B \vee B \mid \neg B \mid x \in S \mid \bot
\end{aligned}
$$

Examples

$$
\begin{aligned}
S_1 \subseteq S_2 &\equiv \forall x.\, x \in S_1 \Rightarrow x \in S_2 \\
S_1 = S_2 \cup S_3 &\equiv \forall x.\, x \in S_1 \Leftrightarrow x \in S_2 \vee x \in S_3 \\
S_1 = S_2 \cap S_3 &\equiv \forall x.\, x \in S_1 \Leftrightarrow x \in S_2 \wedge x \in S_3 \\
S_1 \mid\mid S_2 &\equiv \forall x.\, x \notin S_1 \vee x \notin S_2
\end{aligned}
$$

Evaluating range expressions

$$\begin{aligned}
\lfloor r \rfloor(S, s) &= \lfloor S \rfloor_s \\
\lfloor r \rfloor(R_1 \cup R_2, s) &= \lfloor r \rfloor(R_1, s) \cup \lfloor r \rfloor(R_2, s) \\
\lfloor r \rfloor(R_1 \cap R_2, s) &= \lfloor r \rfloor(R_1, s) \cap \lfloor r \rfloor(R_2, s) \\
\lfloor r \rfloor(\overline{R}, s) &= \overline{\lceil r \rceil(R, s)} \\
\lfloor r \rfloor(\emptyset, s) &= \emptyset
\end{aligned}$$

Transformations

- translate each formula
  into two range expressions per variable

- translate each pair of range expressions
  into code for a propagator

$$p_S(s) \;=\; \langle \lfloor r \rfloor (R_1, s) \cup \lfloor S \rfloor_s, \lceil r \rceil (R_2, s) \cap \lceil S \rceil_s \rangle$$

Properties of transformations

- All projectors are contracting and monotone.
- Every projector is sound for the constraint it implements.
- Translation into projectors is complete with respect to domain-consistency in the approximate domain.

Adding cardinality information

- We can strengthen our domain approximation by adding information about the cardinality of a domain.
- On the one hand, makes modelling more powerful. (Example: dual model of Sudoku)
- On the other hand, domain-consistent propagation of even simple constraints becomes an NP-complete problem.

Cardinality constraint

$$|S| = I$$

$$\begin{array}{rcl}
\top & \implies & |\lfloor S \rfloor| \leq I \\
\top & \implies & I \leq |\lceil S \rceil| \\
n \leq I \ \wedge \ |\lceil S \rceil| = n & \implies & \lceil S \rceil \subseteq S \\
I \leq n \ \wedge \ |\lfloor S \rfloor| = n & \implies & S \subseteq \lfloor S \rfloor
\end{array}$$

The social golfers problem

- Schedule $g \times s$ golfers into
  g groups of s players each over w weeks
  such that no golfer plays in the same group
  with any other golfer more than once.
- An instance of the problem is given by the triple w–g–s.
- Still open: Is there a solution to 10–8–4?

Model

- Represent a week as a list of g set variables, each one with cardinality s.
- Branch such that each player is assigned to all possible groups.

Constraints

- **In each week, each player plays in exactly one group.**
  For any given week, the set of players is partitioned
  by the collection of groups for that week.
- **Each group shares at most one player
  with each other group.**
  The cardinality of the intersection
  of two groups is at most 1.

Best known solution

9–8–4

tournament with 8 groups of 4 golfers each over 9 weeks

```
[ 01 02 03 04 | 05 06 07 08 | 09 10 11 12 | 13 14 15 16 | 17 18 19 20 | 21 22 23 24 | 25 26 27 28 | 29 30 31 32 ]
[ 01 05 09 13 | 02 06 10 14 | 03 07 11 15 | 04 08 12 16 | 17 21 25 29 | 18 22 26 30 | 19 23 27 31 | 20 24 28 32 ]
[ 01 06 11 16 | 02 05 12 15 | 03 08 09 14 | 04 07 10 13 | 17 22 27 32 | 18 21 28 31 | 19 24 25 30 | 20 23 26 29 ]
[ 01 07 17 23 | 02 08 18 24 | 03 05 19 21 | 04 06 20 22 | 09 15 25 31 | 10 16 26 32 | 11 13 27 29 | 12 14 28 30 ]
[ 01 08 19 22 | 02 07 20 21 | 03 06 17 24 | 04 05 18 23 | 09 16 27 30 | 10 15 28 29 | 11 14 25 32 | 12 13 26 31 ]
[ 01 10 18 25 | 02 09 17 26 | 03 12 20 27 | 04 11 19 28 | 05 14 22 29 | 06 13 21 30 | 07 16 24 31 | 08 15 23 32 ]
[ 01 12 21 32 | 02 11 22 31 | 03 10 23 30 | 04 09 24 29 | 05 16 17 28 | 06 15 18 27 | 07 14 19 26 | 08 13 20 25 ]
[ 01 14 20 31 | 02 13 19 32 | 03 16 18 29 | 04 15 17 30 | 05 10 24 27 | 06 09 23 28 | 07 12 22 25 | 08 11 21 26 ]
[ 01 15 24 26 | 02 16 23 25 | 03 13 22 28 | 04 14 21 27 | 05 11 20 30 | 06 12 19 29 | 07 09 18 32 | 08 10 17 31 ]
```

Binary relations

The plan

- use FS constraints to encode binary relations on a fixed (and finite) universe
- express constraints on binary relations as constraints on FS variables

Encoding

- define the notion of the **relational image:**

$$Rx \;=\; \{\, y \in U \mid Rxy \,\}$$

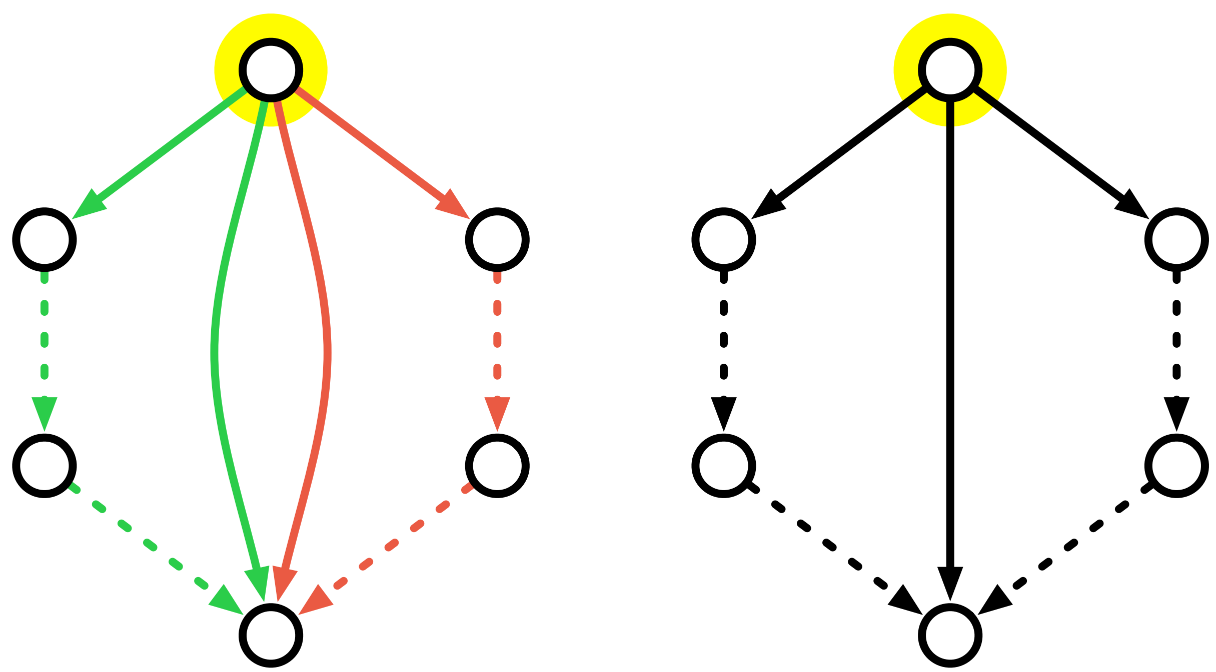
- understand binary relations as total functions from the carrier to subsets of the carrier

$$f_R \;=\; \{\, x \mapsto Rx \mid x \in U \,\}$$

- represent these functions as vectors of finite set variables

Union of two relations

$$A \cup B = C$$

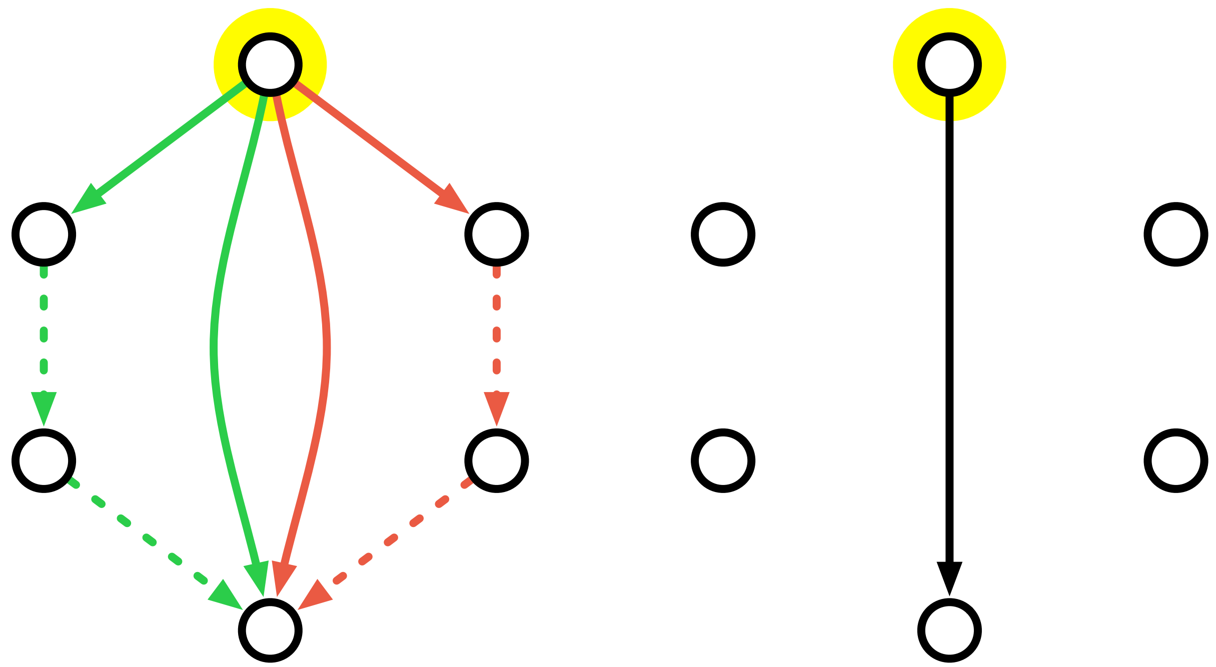


A node j is C-adjacent if and only if j is either A-adjacent to i or B-adjacent to i (or both).

$$\forall i \in [n].\ C\,i = A i \cup B i$$

Intersection of two relations

$$A \cap B = C$$



A node j is C-adjacent to i if and only if j is both A-adjacent to i and B-adjacent to i.

$$\forall i \in [n].\ C\,i = A\,i \cap B\,i$$

Selection constraints

- generalization of binary set operations
- participating elements are variable, too
- example: union with selection

$$S = \bigcup_{i \in S'} S_i$$

- propagation in all directions

Union with selection (1)

$$S = \bigcup \langle S_1, \ldots, S_n \rangle [S']$$

$$\frac{}{S' \subseteq [n]}$$

$$\frac{}{\bigcup_{i \in \lfloor S' \rfloor} \lfloor S_i \rfloor \subseteq S}$$

$$\frac{}{S \subseteq \bigcup_{i \in \lceil S' \rceil} \lceil S_i \rceil}$$

Union with selection (2)

$$S = \bigcup \langle S_1, \ldots, S_n \rangle [S']$$

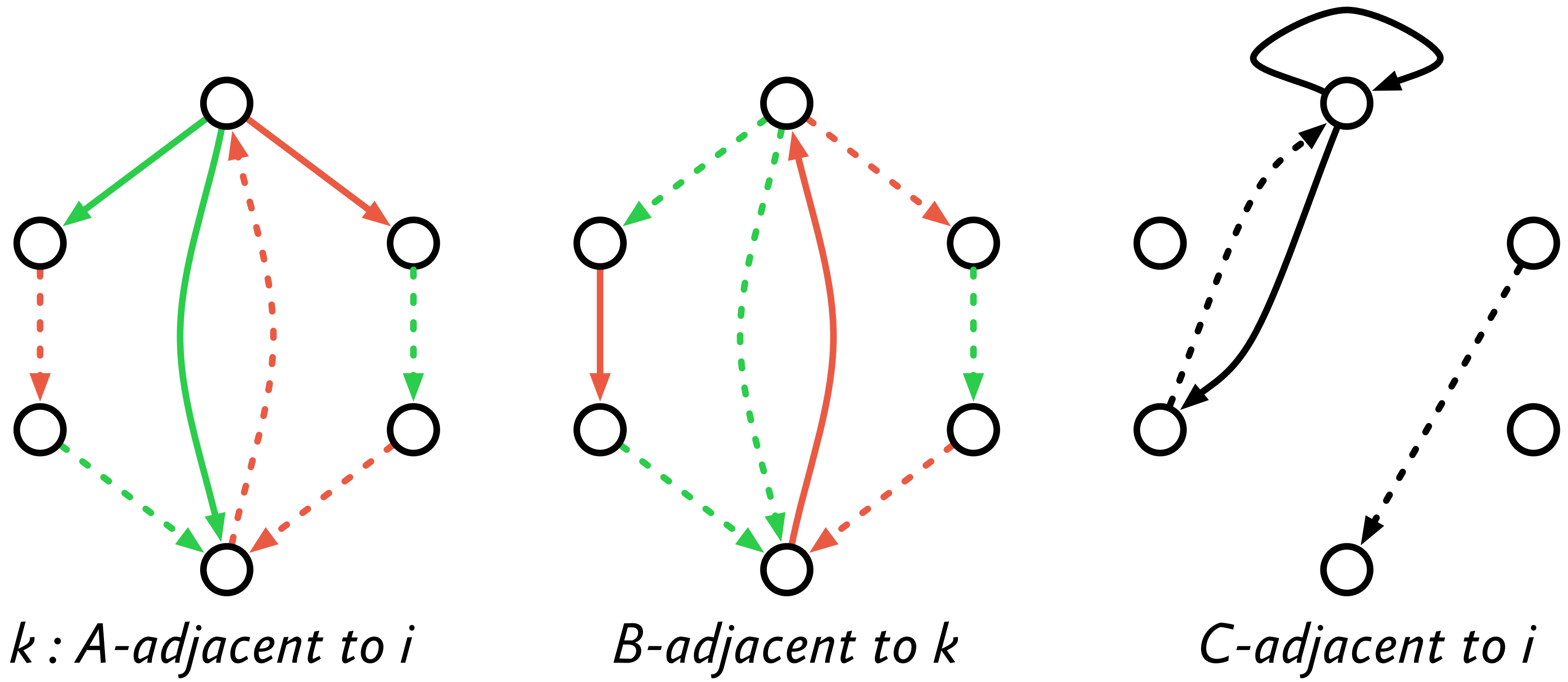
$$\frac{i \in [n] \qquad \lfloor S_i \rfloor \not\subseteq \lceil S \rceil}{i \notin S'}$$

$$\frac{k \in [n] \qquad \lfloor S \rfloor - \bigcup_{i \in \lceil S' \rceil - \{k\}} \lceil S_i \rceil \neq \emptyset}{\lfloor S \rfloor - \bigcup_{i \in \lceil S' \rceil - \{k\}} \lceil S_i \rceil \subseteq S_k}$$

$$\frac{k \in [n] \qquad \lfloor S \rfloor - \bigcup_{i \in \lceil S' \rceil - \{k\}} \lceil S_i \rceil \neq \emptyset}{k \in S'}$$

Composition

$$A \circ B = C$$



A node j is C-adjacent to i if and only if
there exists a node k such that k is A-adjacent to i and j is B-adjacent to k.

$$\forall i \in [n].\, C\,i = \bigcup \langle B1, \ldots, B_n \rangle [A i]$$

Transitivity constraint

$$
\begin{aligned}
R \text{ transitive} \quad &\iff \quad \forall x.\, \forall y.\, \forall z.\, Rxy \wedge Ryz \Rightarrow Rxz \\
&\iff \quad \forall x.\, \forall y \in Rx.\, \forall z \in Ry.\, Rxz \\
&\iff \quad \forall x.\, \forall y \in Rx.\, \forall z \in Ry.\, z \in Rx \\
&\iff \quad \forall x.\, \forall y \in Rx.\, Ry \subseteq Rx \\
&\iff \quad \forall x.\, \left(\bigcup_{y \in Rx} Ry \right) \subseteq Rx
\end{aligned}
$$

Transitivity constraint

$$
\begin{aligned}
R \text{ transitive} \quad &\iff \quad \forall x.\, \forall y.\, \forall z.\, Rxy \wedge Ryz \Rightarrow Rxz \\
&\iff \quad \forall x.\, \forall y \in Rx.\, \forall z \in Ry.\, Rxz \\
&\iff \quad \forall x.\, \forall y \in Rx.\, \forall z \in Ry.\, z \in Rx \\
&\iff \quad \forall x.\, \forall y \in Rx.\, Ry \subseteq Rx \\
&\iff \quad \forall x.\, \left(\bigcup_{y \in Rx} Ry \right) \subseteq Rx
\end{aligned}
$$

selection constraint

Summing up

- used vectors of FS variables
  to encode binary relations
- constraints on binary relations
  can be stated as constraints on FS variables
- featured on next assignment

Fourth graded assignment

- alternative model for Sudoku
- implement a structure for constraints on binary relations
- implement solvers for rooted trees:
    - unordered trees
    - ordered trees