Software Verification and Testing

Lecture Notes: Sets, Relations, Functions

Sets

problem: FOL is still not ideal for specifying and verifying software systems

- too unexpressive: no quantification over functions or predicates
- too unspecific: special functions, predicates, types are needed
- too concrete: abstract structural properties cannot be expressed directly

examples: we would like to

1. model programs as functions on sets of locations; characterise all programs or some programs with a given property
2. express temporal properties of programs (safety, deadlock, starvation,. . .)
3. model data structures and data types
4. type programs

solutions: we use set theory and later temporal logics

Sets

example:

natural language:	All horses are animals
FOL:	$\forall x. isHorse(x) \rightarrow isAnimal(x)$
set theory:	$\forall x. x \in Horses \rightarrow x \in Animals$
	$Horses \subseteq Animals$

natural language:	All horses' heads are animals' heads
FOL:	$\forall x. isHorse(x) \wedge isHead(x) \rightarrow isAnimal(x) \wedge isHead(x)$
set theory:	$Horses \cap Heads \subseteq Animals \cap Heads$

$Horses \subseteq Animals \rightarrow Horses \cap Heads \subseteq Animals \cap Heads$
follows immediately from isotonicity of intersection (see later). . .

Sets

set theory:

- in mathematics: universal language and tool
- in software engineering: basis of formal methods such as Z or B

axiomatic set theory:

- complex formalism belonging to foundations of mathematics
- many axioms needed to circumvent paradoxes:
  the set of all sets that do not contain itself as an element (B. Russell)

operational set theory:

- first-order set theory with types to avoid paradoxes
- here: we use set theory only as a specification language

Sets

language of set theory: FOL with distinguished binary predicate symbol $\in$

properties of set:

- extensionality: two sets are equal if they have the same elements

$$\forall x, y.(\forall z.(z \in x \leftrightarrow z \in y) \rightarrow x = y)$$

- comprehension: the elements for which a predicate ϕ holds form a set

$$\{x : \phi\}$$

Sets

properties of set:

- closure under pairs: the pair of two sets x and y is the set

$$x \times y = \{(a, b) : a \in x \wedge b \in y\}$$

  pairs can be defined within set theory. . .
- existence of power sets: the power set of a set x is the set

$$2^x = \{y : \forall z. z \in y \rightarrow z \in x\}$$

Sets

set inclusion: $\forall x, y. x \subseteq y \leftrightarrow \forall z. z \in x \rightarrow z \in y$

empty set: $\emptyset = \{z : z \neq z\}$

set union: $x \cup y = \{z : z \in x \vee z \in y\}$

set intersection: $x \cap y = \{z : z \in x \wedge z \in y\}$

set complementation: $\overline{x} = \{z : z \notin x\}$

equality of pairs: $\forall x, y, x', y'. (x, y) = (x', y') \leftrightarrow x = x' \wedge y = y'$

power set and inclusion: $\forall x, y. x \in 2^y \leftrightarrow x \subseteq y$

Sets

theorem: let A be a set. Then $(2^A, \cup, \cap, ^-, A, \emptyset)$ is a Boolean algebra

consequences: for $x, y, z \subseteq A$ we have the following algebraic properties

- associativity: $x \cup (y \cup z) = (x \cup y) \cup z \quad x \cap (y \cap z) = (x \cap y) \cap z$
- commutativity: $x \cup y = y \cup x \quad x \cap y = y \cap x$
- distributivity: $x \cup (y \cap z) = (x \cup y) \cap (x \cup z) \quad x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$
- absorption: $x \cup (y \cap x) = x = x \cap (y \cup z)$
- idempotence: $x \cup x = x = x \cap x$
- zero: $x \cup \emptyset = x \quad x \cap \emptyset = \emptyset$
- unit: $x \cup s = s \quad x \cap s = x$
- complement: $x \cup \overline{x} = s \quad x \cap \overline{x} = \emptyset \quad \overline{\overline{x}} = x$
- de Morgan: $x \cup y = \overline{\overline{x} \cap \overline{y}} \quad x \cap y = \overline{\overline{x} \cup \overline{y}}$
- isotonicity: $x \subseteq y \rightarrow x \cup z \subseteq y \cup z \quad x \subseteq y \rightarrow x \cap z \subseteq y \cap z$
- antitonicity: $x \subseteq y \rightarrow \overline{y} \subseteq \overline{x}$

Sets

more consequences: for $x, y, z \subseteq A$

$$x \cup y \subseteq z \leftrightarrow x \subseteq z \wedge y \subseteq z \qquad x \subseteq y \cap z \leftrightarrow x \subseteq y \wedge x \subseteq z$$

further operations:

- set difference: $x - y = x \cap \overline{y}$
- exclusive or: $x + y = (x - y) \cup (y - x) = (x \cup y) \cap \overline{x \cap y}$

supremum and infimum: Let $B \subseteq 2^A$ be some set of sets

$$\sup(B) = \{\text{least subset of } A \text{ that contains all elements of } B\}$$

$$\inf(B) = \{\text{greatest subset of } A \text{ that is contained in all elements of } A\}$$

Sets

conclusion: the algebra of set allows very concise abstract reasoning

problem: this approach to set theory is inconsistent.
Consider $A = \{x : x \notin x\}$. Then $A \in A \leftrightarrow A \notin A$

intuition: sets should be constructed from other sets

solutions:

- foundational: modify axioms, restrict comprehension
- operational: add types to set

here: we do not treat this...

Binary Relations

definition: Let A be a set. A binary relation R on A is a subset of $A \times A$

example:

$$\begin{aligned}
Osbournes &= \{Sharon, Ozzy, Kelly, Jack\} \\
men &= \{Ozzy, Jack\} \\
women &= \{Sharon, Kelly\} \\
parent &= \{(Sharon, Kelly), (Sharon, Jack), (Ozzy, Kelly), (Ozzy, Jack)\} \\
mother &= \{(Sharon, Kelly), (Sharon, Jack)\} \\
son &= \{(Jack, Ozzy), (Jack, Sharon)\} \\
sibling &= \{(Jack, Kelly), (Kelly, Jack)\}
\end{aligned}$$

Binary Relations

representations: every finite relation can be represented as a

- $0 - 1$-matrix

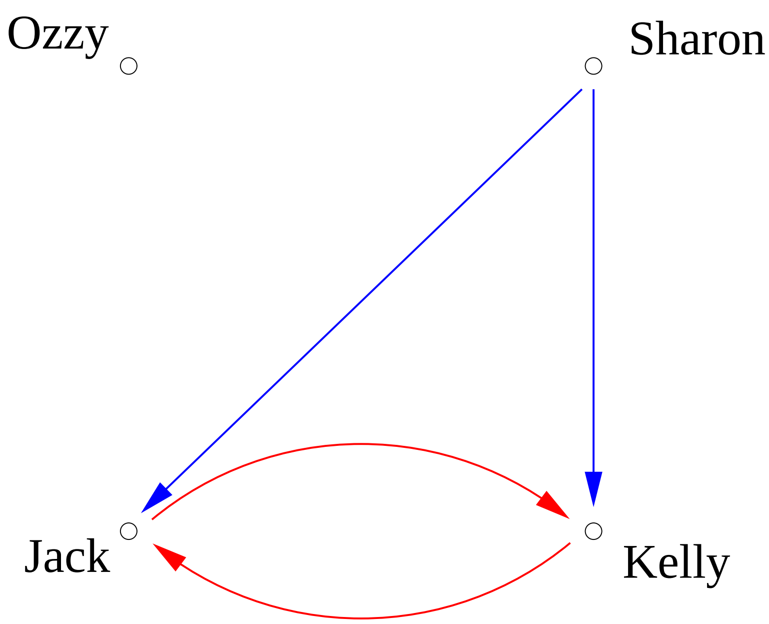
$mother$	S	O	K	J
S	0	0	1	1
O	0	0	0	0
K	0	0	0	0
J	0	0	0	0

$$mother = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Binary Relations

representations: every finite relation can be represented as a

- directed graph (digraph) $G = (V, E)$ with finite set of vertices V and set of edges $E \subseteq V \times V$



Binary Relations

operations on relations: consider the set of relations on a set A

- as sets, they form a boolean algebra with maximal element $A \times A$
- the identity relation

$$1_A = \{(x, x) : x \in A\}$$

- the converse of a relation R is

$$R^\circ = \{(y, x) : (x, y) \in R\}$$

- the product of two relations R and S is

$$R \circ S = \{(x, y) : \exists z.(x, z) \in R \wedge (z, y) \in S\}$$

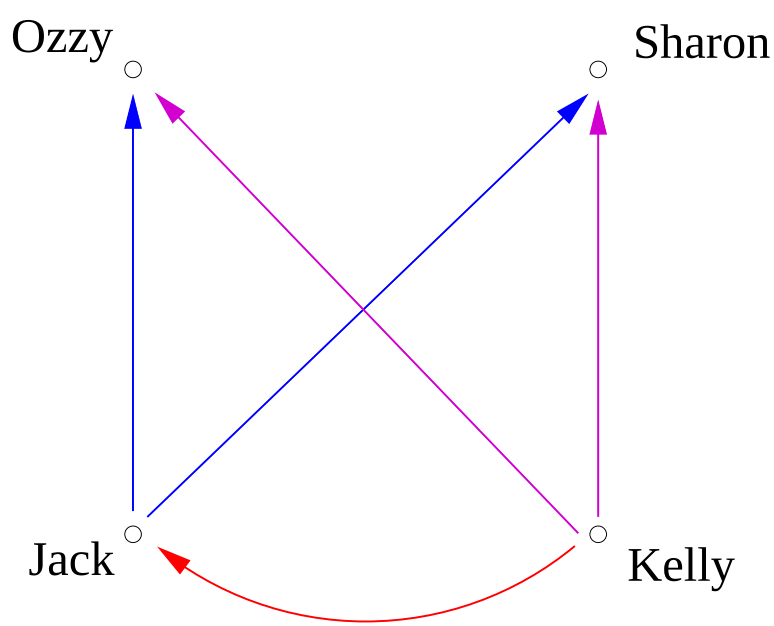
Binary Relations

example: analysing the Osbournes

$$\begin{aligned}
parent - mother &= \{(Ozzy, Kelly), (Ozzy, Jack)\} \\
&= father \\
parent^{\circ} &= \{(Kelly, Sharon), (Jack, Sharon), \\
&\quad\ (Kelly, Ozzy), (Jack, Ozzy)\} \\
&= child \\
sibling^{\circ} &= sibling \\
sibling \circ son &= \{(Kelly, Sharon), (Kelly, Ozzy)\} \\
&= daughter
\end{aligned}$$

Binary Relations

example: analysing the Osbournes



Binary Relations

algebra of relations: relations satisfy again many algebraic laws

examples:

$$(R^\circ)^\circ = R \qquad (R \cup S)^\circ = R^\circ \cup S^\circ \qquad (R \circ S)^\circ = S^\circ \circ R^\circ$$

$$(R \circ S) \circ T = R \circ (S \circ T) \qquad R \circ (S \cup T) = R \circ S \cup R \circ T$$

$$1_A \circ R = R \circ 1_A \qquad \emptyset \circ R = \emptyset = R \circ \emptyset$$

Binary Relations

properties of relations: R is

- reflexive iff $1_A \subseteq R$ iff $\forall x.(x,x) \in R$
- symmetric iff $R^\circ \subseteq R$ iff $\forall x,y.(x,y) \in R \leftrightarrow (y,x) \in R$
- anti-symmetric iff $R \cap R^\circ \subseteq 1_A$ iff $\forall x,y.(x,y) \in R \wedge (y,x) \in R \to x = y$
- transitive iff $R \circ R \subseteq R$ iff $\forall x,y,z.(x,y) \in R \wedge (y,z) \in R \to (x,z) \in R$

definition: a reflexive antisymmetric, transitive relation is a partial order

definition: a reflexive symmetric, transitive relation is an equivalence relation

Binary Relations

examples:

- $\subseteq$ is a partial ordering
- $=$ is an equivalence relation
- $\rightarrow$ is reflexive and transitive, but not antisymmetric:
  $p \wedge q \rightarrow q \wedge p$ and $q \wedge p \rightarrow p \wedge q$, but $p \wedge q \neq q \wedge p$

Binary Relations

definition: we inductively define $R^0 = 1_A$ and $R^{n+1} = R \circ R^n$

definition:

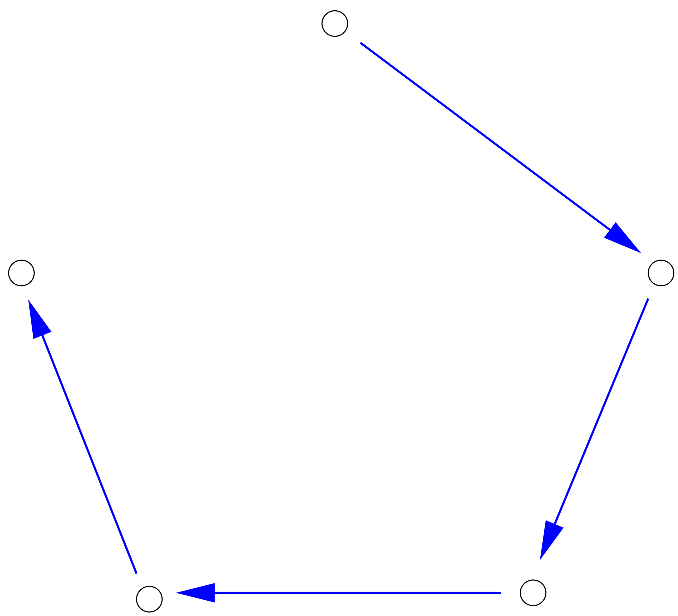
- the transitive closure of R is $R^+ = \sup(R^i : i > 0)$
- the reflexive transitive closure of R is $R^* = \sup(R^i : i \geq 0)$

remark:

- R^+ is the least transitive relation containing R
- R^* is the least reflexive transitive relation containing R

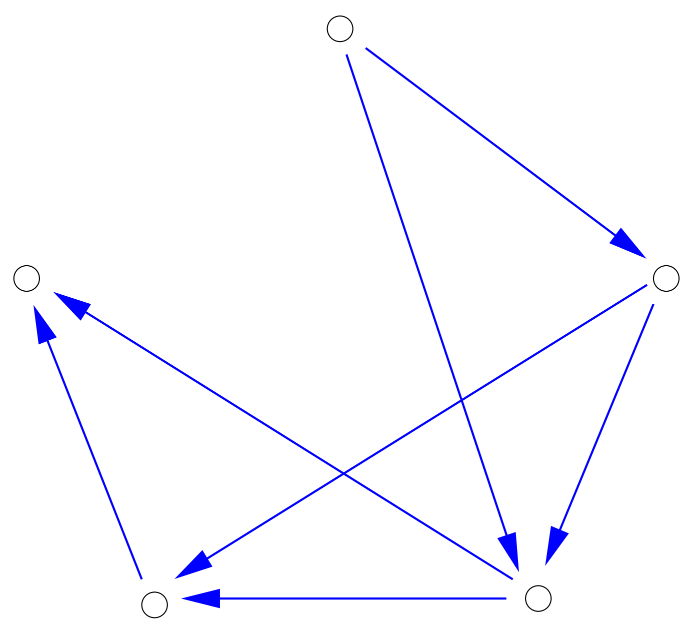
Binary Relations

example: transitive closure



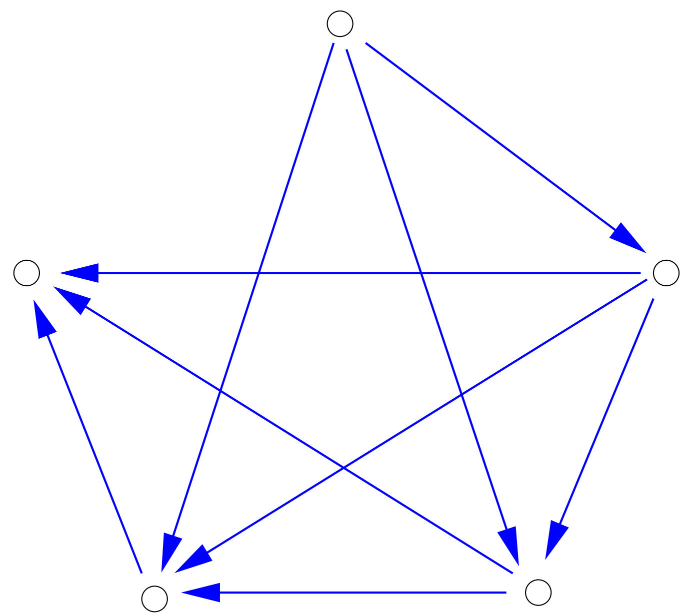
Binary Relations

example: transitive closure



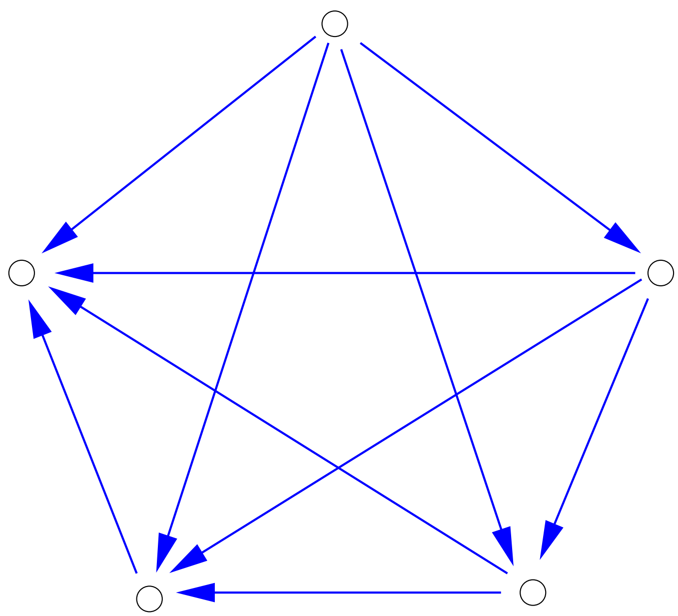
Binary Relations

example: transitive closure



Binary Relations

example: transitive closure



Binary Relations

domain related constructs: Let $R \subseteq S \times S$ and $P \subseteq S$

- domain: $dom(R) = \{x : (x, y) \in R\}$
- range: $ran(R) = \{y : (x, y) \in R\}$
- domain restriction: $P \lhd R = \{(p, y) : p \in P \wedge (p, y) \in R\}$
- range restriction: $R \rhd P = \{(x, p) : p \in P \wedge (x, p) \in R\}$
- domain subtraction: $P \mathbin{\triangleleft\!\!\!-} R = (dom(R) - P) \lhd R$
- range subtraction: $R \mathbin{\triangleright\!\!\!-} P = R \rhd (ran(R) - S)$
- preimage: $|R\rangle P = dom(R \rhd P)$
- image: $\langle R|P = ran(P \lhd R)$

Binary Relations

examples:

$$
\begin{aligned}
dom(parent) &= \{Sharon, Ozzy\} \\
ran(parent) &= \{Kelly, Jack\} \\
mother \rhd \{Kelly\} &= \{(Sharon, Kelly)\} \\
mother \unrhd \{Kelly\} &= \{(Sharon, Jack)\} \\
|mother\rangle\{Kelly, Jack\} &= \{Sharon\} \\
\langle son|\{Jack\} &= \{Sharon, Ozzy\}
\end{aligned}
$$

can you visualise this using graphs?

Binary Relations

further constructs: Let $R \subseteq S \times S$ and $P \subseteq S$

- overwriting: $R \mathbin{\lhd\!\!\!+} S = dom(S) \mathbin{\lhd\!\!\!-} R \cup S$
- direct product: $R \otimes S = \{(x, (y, z)) : (x, y) \in R \wedge (x, z) \in S\}$
- parallel product: $R || S = \{((x_1, x_2), (y_1, y_2)) : (x_1, y_1) \in R \wedge (x_2, y_2) \in S\}$

example:

$$sibling \mathbin{\lhd\!\!\!+} son = \{(Jack, Ozzy), (Jack, Kelly), (Kelly, Jack)\}$$

Binary Relations

problem: we have only considered homogenous relations on one single set A

definition: A heterogeneous relation R between sets A and B is a subset of $A \times B$

advantage: more precise specifications

remark: the previous calculus extends to heterogeneous relations

- the boolean operations and conversion are straightforward
- relational products must respect the sets, i.e., $R \circ S$ is only defined for $R \subseteq A \times B$ and $S \subseteq B \times C$

notation: we write $A \leftrightarrow B$ for the set of all relations from A to B

Functions

idea: a function is a heterogeneous relation where each element in the domain is related to precisely one element in the range

definition: $f \subseteq A \times B$ is

- partial function if $f^\circ \circ f \subseteq 1_B$
- total if $1_A \subseteq f \circ f^\circ$ iff $dom(f) = A$
- surjective if $1_B \subseteq f^\circ \circ f$ iff $ran(f) = B$
- injective if $f \circ f^\circ \subseteq 1_A$
- bijective if it is injective and surjective

Functions

notation: we write

- $A \dot{\to} B$ for the set of partial functions from A to B
- $A \longrightarrow B$ for the set of total functions
- $A \rightarrowtail B$ for the set of injections
- $A \twoheadrightarrow B$ for the set of surjections
- $A \rightarrowtail\!\!\!\!\twoheadrightarrow B$ for the set of bijections

Functions

intuition:

- if f is a partial function then $(x, y) \in f$ and $(x, z) \in f$ imply $y = z$, whence $f^\circ \circ f \subseteq 1_A$
- if f is total then it is defined everywhere on A
- if f is surjective then its range is B, whence its converse is total
- if f is injective, then $x \neq y$ implies $f(x) \neq f(y)$, whence f° is a partial function

remark: the constructions on relations and functions lead to a huge set of algebraic properties (see Abrial's book)

Functions

example: kinship relations (very strict society)

- constraints:
  - every person is either male or female, but not both
  - only women have husbands and at most one
  - only men have wives and at most one
  - mothers are married women

Functions

example: kinship relations

- fundamental set: $People$
- constraints formalised:

$$
\begin{aligned}
women &\subseteq People \\
men &= People - women \\
husband &\in women \dot{\rightarrowtail} men \\
mother &\in People \dot{\rightarrow} dom(husband)
\end{aligned}
$$

Functions

example: kinship relations

- derived concepts:

$$
\begin{aligned}
\mathit{wife} &= \mathit{husband}^{\circ} & \mathit{spouse} &= \mathit{husband} \cup \mathit{wife} \\
\mathit{father} &= \mathit{mother} \circ \mathit{husband} & \mathit{parents} &= \mathit{mother} \otimes \mathit{father} \\
\mathit{children} &= (\mathit{mother} \cup \mathit{father})^{\circ} & \mathit{daughter} &= \mathit{children} \rhd \mathit{women} \\
\mathit{sibling} &= \mathit{children}^{\circ} \circ \mathit{children} - 1_{\mathit{People}} & \mathit{brother} &= \mathit{sibling} \mathbin{⩥} \mathit{women}
\end{aligned}
$$

Functions

example: kinship relations

- properties:

$$
\begin{aligned}
mother &= father \circ wife & spouse &= spouse^\circ \\
ran(parents) &= husband & sibling &= sibling^\circ \\
father \circ father^\circ &= mother \circ mother^\circ & father \circ mother^\circ &= \emptyset \\
mother \circ father^\circ &= \emptyset & father \circ children &= mother \circ children
\end{aligned}
$$

question: can you prove these?

Functions

example proof: kinship relations

$$\begin{aligned}
father \circ father^\circ &= (mother \circ husband) \circ (mother \circ husband)^\circ \\
&= mother \circ husband \circ husband^\circ \circ mother^\circ \\
&= mother \circ 1_{dom(husband)} \circ mother^\circ \\
&= mother \rhd dom(husband) \circ mother^\circ \\
&= mother \circ mother^\circ
\end{aligned}$$

Functions

example: regular programs on state space A

- model actions of a program by relations on A
- model tests by subsets of A
- empty action: skip $= 1_A$
- abortive action: abort $= \emptyset$
- sequencing: $R; S = R \circ S$
- non-determinism: $R + S = R \cup S$
- conditional: if B then R else $S = B \lhd R \cup (A - B) \lhd S$
- loop: while B do $R = (B \lhd R)^* \circ (A - B)$

Conclusion

relational calculus:

- builds two layers of abstraction on logic
- very suitable tool for specifying properties of systems
- abstract verifications in equational calculus
- huge libraries of rules for various constructs difficult to manipulate

outlook: many concepts will reappear in Z