Functional Requirements: Formal Structural Requirements

- Why Formal? – Revisiting SADT
- RML/Telos Essentials
- A Formalization of RML/Telos
- A Survey of Formal Methods

SADT Revisited

- SADT: ambiguities persist in narrative
  - ***Boxes inside a box may represent***

    specializations: e.g., A1 isA A0

    instances: e.g., A1 instanceOf A0

    aggregation: e.g., A1 partOf A0

    of the concept represented by the box
  - ***Temporal relationships are not clear:***

    When are inputs produced?

    as a chunk, in a piece-meal, upon request

    When are outputs produced?

    immediately after receiving inputs, anytime, according to controls

    When do boxes perform actions?

    sequentially, concurrently throughout, partially overlapping
  - ***Attributes of data not easy to express:***

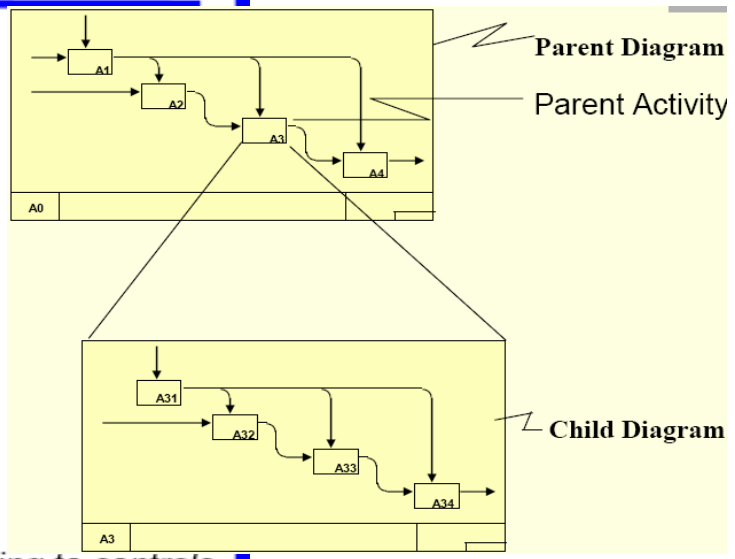
    How do we aggreate {name, age, address, ...} of a person?
  - ***Constraints on arrows now easy to express:***

    Are some inputs optional/mandatory?

    Are any inputs legal?

    What is the difference between inputs and outputs?

    Are controls always clearly different from mechanisms?



Lawrence Chung

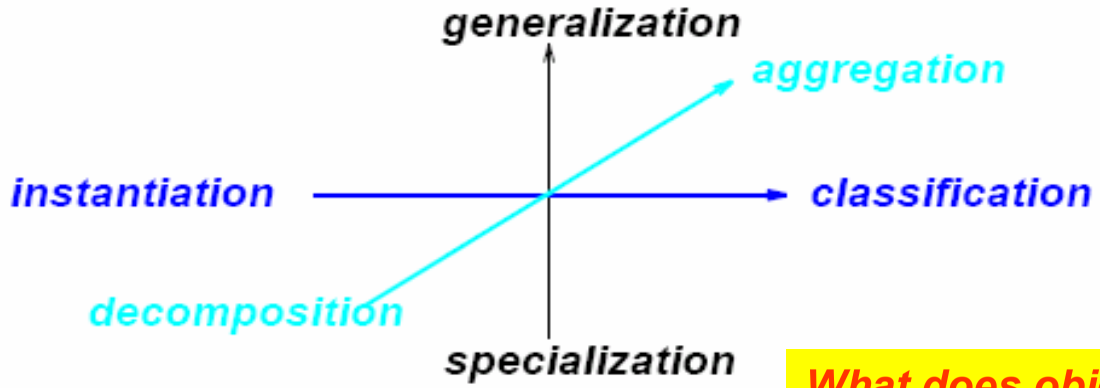
RML/Telos Essentials

[S. Greenspan, J. Mylopoulos and A. Borgida, A. On formal requirements modeling languages: RML revisited, 16th Int. Conf. on Software Eng. pp135 -147. 1994.

* Ontological primitives:

  entities: objects in the domain of discourse
  activities: induce changes in the world
  assertions: constraints on the behavior of entities and activities

* epistemological (abstraction/structural) primitives:



What does object-orientation mean?

V ***aggregation/decomposition:*** ***categories of attributes***
relationships between entities/objects are treated also as objects

Example1:

EntityClass Person
familyDoctor: MD
when-created: Date
when-terminated: Date

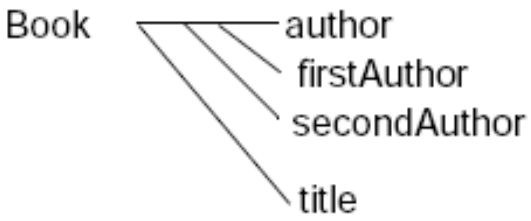
In philosophy, ontology is the study of being or existence. It seeks to describe or posit the basic categories and relationships of being or existence to define entities and types of entities within its framework. Ontology can be said to study conceptions of reality. http://en.wikipedia.org/wiki/Ontology

RML/Telos Essentials

- aggregation/decomposition: categories of attributes

relationships between entities/objects are treated also as objects

Example2:



EntityClass Book
  author
    firstAuthor: Person
    secondAuthor: Person
  title: string

- generalization/specialization: ?

infinite hierarchy of classes

tokens, classes, metaclasses, metametaclasses, ... omegaclass

Recall:

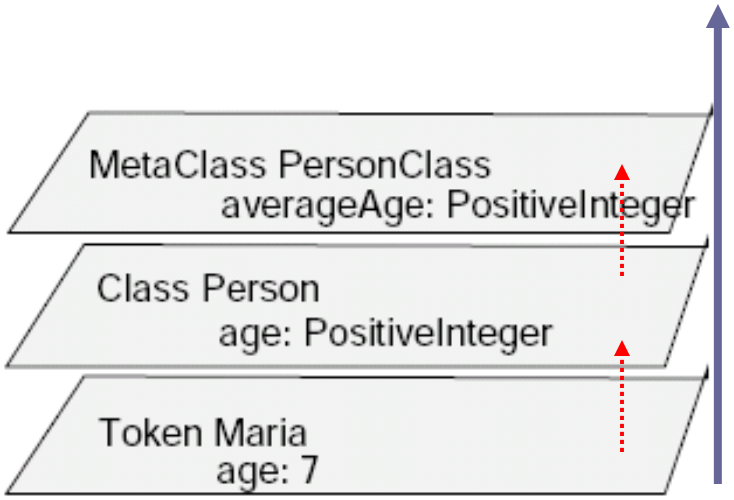
an object is a "concept" of anything (inclusing concept)

each class is a concept

each class is an object

each object belongs to a class

each class belongs to some class!!



cf. Liar's paradox;
Russell's paradox:

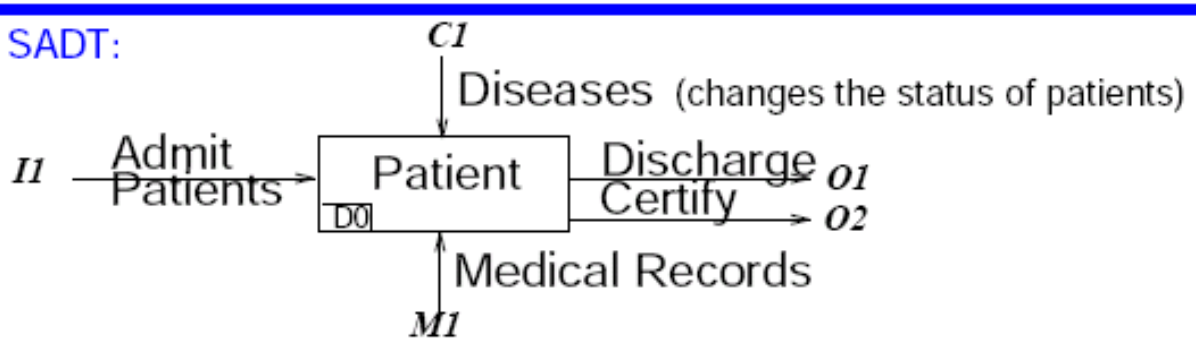
A is an element of *M* if and only if *A* is not an element of *A*

$M = \{A \mid A \notin A\}.$ is *M* an element of itself?

[A. Church, *Theory of Types*, 1940.]

Entities

* SADT:



* OO-RML

```
EntityClass Persons with
  necessary part
      name: Names
      gender: {'Male, 'Female}
  association
      address: Addresses
      nearestRelative: Persons
      familyDoctor: MD
          when-created: Date
          when-terminated: Date
      favoriteBook: Book
```

```
EntityClass Patients with
  necessary unique part
      record: Medical-Records
  association
      location: Nursing-Homes
      room: Rooms
      Physician: MD
      diagnosis: Diseases
      paymentDue: $Values
  producer
      register: AdmitPatients (person=this, toHome=loc)
  consumer
      release: Discharge (patient=this)
      decease: Certify (certifee=this, status=dead)
  initially
      startClean?: (paymentDue=0)
```

; *part: properties which express relationships which ordinarily do not change with time*

; *association: the property value may change over time*

; *necessary: attribute each instance should have always*

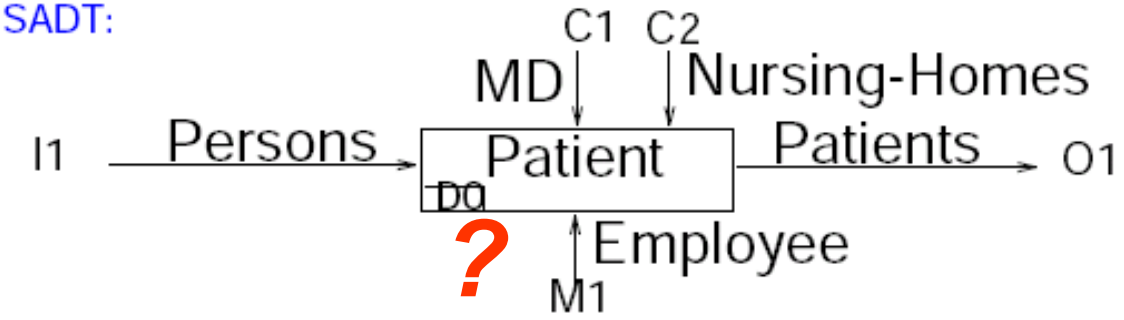
; *producer: the property value is an event, one of whose effects is to make a new instance of the class being defined*

; *consumer: the property value is an event one of whose effects is to make an instance of the class being defined stop being an instance of the class*

Uniform treatment of attributes and associations

Activities

* SADT:



* OO-RML

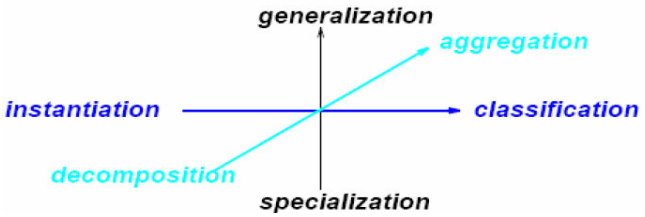
```
ActivityClass AdmitPatients with
   input
       person:Persons
   control                                  ;  activation condition, termination condition
       toHome: Nursing-Homes                   when should an activity start and end
       doc: MD
   mechanism                                ;  initially (preconditions) and finally (postconditions):
       cleark: Employee                        what conditions should hold for an activity to start;
   output                                      what conditions should hold at the time of termination
       patient: Patients
   initially
       already-in?: not (person in Patients)
   finally
       admitted?: (person=patient) and (patient.location=toHome)
   part
       getBasicInfo: Interview (whom=person)
       place: AssignRoom (toWhom=person)
       getConsult: ScheduleVisit (visitor=doc, visitee=patient)
       assess: TakeVitalSigns (visitee=patient)
   integrityConstraint
```

RML/Telos – Assertion Class

```
AssertionClass IsTreatedWith with
        arguments
                p: Patients
                t: Treatments
        necessary
                cond1: Available (treatment = t, at = p.location)
                cond2: Recommended (treatment = t, disease = p.diagnosis)
end IsTreatedWith

AssertionClass Available with
        arguments
                treatment: Treatments
                at: Nursing-Home
end Available

AssertionClass ReceiveChemotherapy isA IsTreatedWith with
        arguments
                p: CancerPatients
                t: CancerDrugs
```



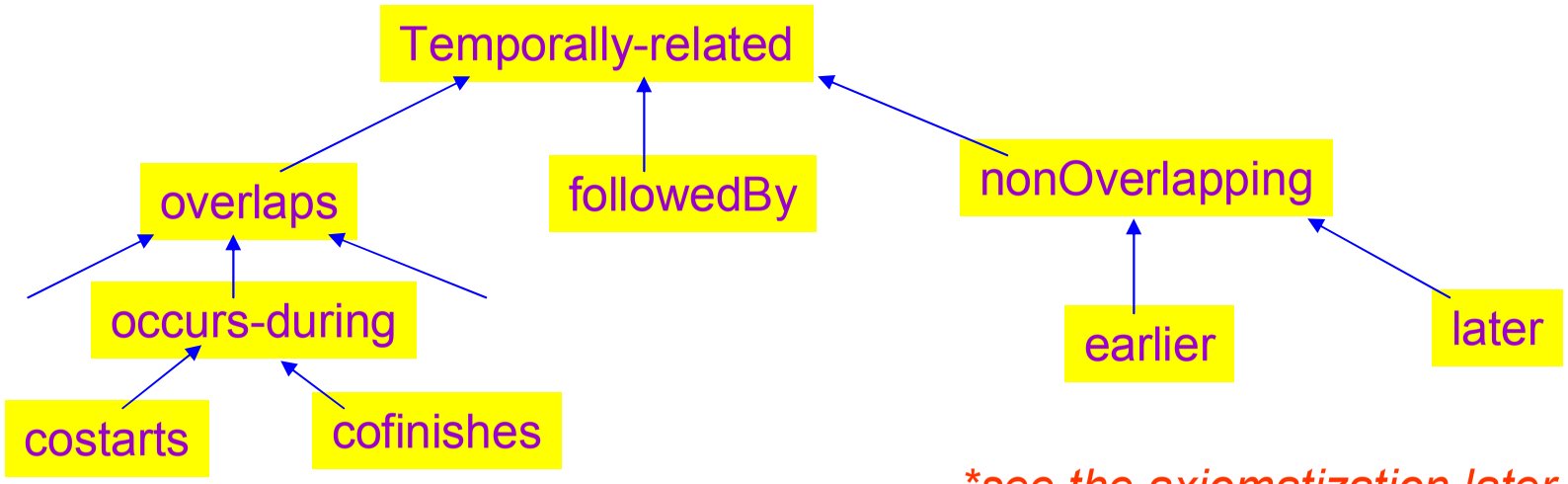
What could be instances?

RML/Telos – MetaClass

```
MetaClass ActivityClass with
        input: EntityClass
        output: EntityClass
        control: EntityClass
        part: ActivityClass
end ActivityClass

ActivityClass AdmitPatients with
        input
                : Person
        output
                : Patient
        control
                : MedicalDoctor
        part
                getBasicInfo: Interview (whom = person)
                place: AssignRoom (toWhom = person)
                getConsult: ScheduleVisit (visitor = doc, visitee = patient)
                assess: takeVitalSigns (visitee = patient)
```

RML/Telos – Temporal Primitives



*see the axiomatization later on

```
ActivityClass AdmitPatients with
        …
        part
                getBasicInfo: Interview (whom = person)
                place: AssignRoom (toWhom = person)
                getConsult: ScheduleVisit (visitor = doc, visitee = patient)
                assess: takeVitalSigns (visitee = patient)
        constraints
                getBasicInfo followedBy place
                place overlaps getConsult
```

A Formalization of RML/Telos

[S. Greenspan, J. Mylopoulos and A. Borgida, A. “A Requirements Modeling Language and Its Logic,”
In M. Brodie and J. Mylopoulos (eds), *On Knowledge Base Management Systems*, Springer-Verlag. 1986. pp 471 -502.]

Formal Semantics

- formal = syntax = grammar
  - *proof theory/axiomatics* for deriving new statements from old

  => reasoning, consistency checking, derive new facts
  - deductive reasoning
  - abductive reasoning
  - approximate reasoning
  - analogical reasoning
  - qualitative reasoning
  - probabilistic reasoning
  - …

- semantics: meaning
  - *model theory/semantics* for making sense of them

  - Denotational semantics,
  - Tarskian semantics,
  - Kripke's possible world semantics,
  - Hoare's partial correctness semantics
  - …

  Can be in the form of
  - semantic integrity constraints/assertions, rules)

Logic: Brief Review

- *formal language* for expressing statements
- *model theory/semantics* for making sense of them
- *proof theory/axiomatics* for deriving new statements from old

❖ connectives:

- ^, & logical conjunction
- ∨ logical disjunction
- →, ⊃ material implication (implies; if .. Then)
- ↔, ≡ material equivalence (if and only if; iff)
- ¬, ~ logical negation

❖ quantifiers:

- ∀ universal quantifier
- ∃ existential quantifier

❖ modality:

- □ necessity
- ◇ possibility

❖ meta logical properties

- ⊨ semantic consequence
- ⊢ syntactic consequence

Propositional Logic: Proof Theory

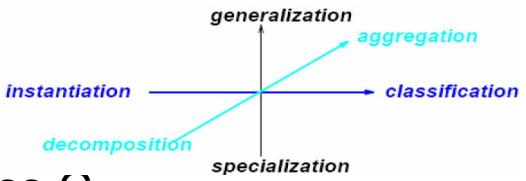
- A **proof Theory** $\mathcal{P}$ defines **derivations** (proofs) in terms of **axioms** and **rules of inference**
- A **derivation** is a finite sequence of formulas, in which each formula is an **axiom** in $\mathcal{P}$, a **premise**, or follows from earlier formulas in the sequence by a rule of inference in $\mathcal{P}$
- B is **derivable** from $A_1, \ldots, A_n$ in P, written $A_1, \ldots, A_n \vdash_P B$ if there is a derivation for B in $\mathcal{P}$ with premises $A_1, \ldots, A_n$
- A proof theory $\mathcal{P}$ is **sound** if $A \vdash_P B$ implies $A \models B$ — *Every derivable formula is valid.* ($\mathcal{I}$(A) ⊨ $\mathcal{I}$(B))
- A proof theory $\mathcal{P}$ is **complete** if $A \models B$ implies $A \vdash_P B$ — *Every valid formula is derivable.*

[H. Geffner, http://www.tecn.upf.es/~hgeffner/html/cursos/slides/logica.pdf]

For more, see www.voronkov.com/slides/brics01.ps

logical consistency: two statements are ***consistent*** if and only if their conjunction is not a contradiction

Formal Semantics of RML/Telos



❖Basic Predicates

❖IN (i, C, s): predicate asserting that token I is an instance of class C

E.g., IN (John, Child, 1997)
IN (John, Student, 2004)

❖ IS-A (C_1, C_2): time-independent predicate asserting that class C_1 is a subclass of class C_2.

E.g., IS-A (Child, Person)
IS-A (Student, Person)

❖ PROPDEF (C, a): time-independent function which gives the class to which the value of attribute a for instances of class C must belong (definitional property function)

E.g., PROPDEF (Person, age) => PersonAgeValue
where Class PersonAgeValue = 0..200
PROPDEF (Child, age) => ChildAgeValue
where Class ChildAgeValue = 1..17

Formal Semantics of RML/Telos

❖Some Axioms

❖ Partial Order IS-A Constraint

❑ reflexive (C, C):
E.g., IS-A (Person, Person)
IS-A (Child, Child)

❑ anti-symmetric: $(C_1 \neq C_2) \wedge$ IS-A $(C_1, C_2) \leftrightarrow \sim$ IS-A (C_2, C_1)
E.g., IS-A (Child, Person) $\rightarrow \sim$ IS-A (Child, Person)

❑ transitive: IS-A $(C_1, C_2) \wedge$ IS-A $(C_2, C_3) \rightarrow$ IS-A (C_1, C_3)
E.g., IS-A (ChildStudent, Child) $\wedge$ IS-A (Child, Person)
$\rightarrow$ IS-A (ChildStudent, Person)

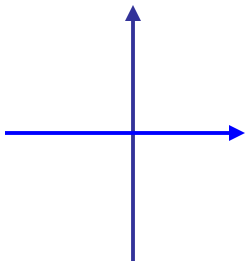
- What are the subclasses of class C?
- What are the superclasses of class C?
- Are classes C_1 and C_2 related to each other?
- What inconsistencies are there?
  E.g., IS-A (C_2, C_3) IS-A (C_1, C_2) IS-A (C_3, C_1)

Formal Semantics of RML/Telos

❖Axioms

❖ Extensional IS-A Constraint

❑ IN (I, C_1, s) $\wedge$ IS-A (C_1, C_2) $\rightarrow$ IN (I, C_2, s)



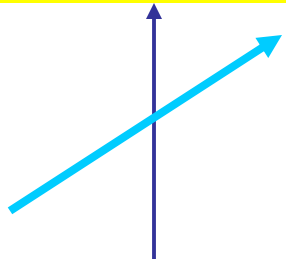
Each instance of a class is in its superclass

- IN (Cathy, Child, 2004) $\wedge$ IS-A (Child, Person) $\rightarrow$ IN (cathy, Person, 2004)?
- IN (Maria, Person, 2004) $\wedge$ IS-A (Child, Person) $\rightarrow$ IN (Maria, Child, 2004)?
- IN (Chris, Child, 2004) $\wedge$ IS-A (Child, Person) $\rightarrow$ IN (Chris, Person, 2005)?
- IN (Person, PersonClass, 1996) $\wedge$ IN (John, Person, 2004)
  $\rightarrow$ IN (John, PersonClass, 2004)?

Formal Semantics of RML/Telos

❖Axioms

Uniform treatment of attributes and associations



❖ Intensional IS-A Constraint

❑ PROPDEF (C_2, a) = $A_2 \wedge$ IS-A (C_1, C_2)
$\exists A_1$ [PROPDEF (C_1, a) = $A_1 \wedge$ IS-A (A_1, A_2)]

Inherited attributes have more specialized ranges

➢ PROPDEF (Person, age) = PersonAge Value $\wedge$ IS-A (Child, Person)
$\exists A_1$ [PROPDEF (Child, age) = $A_1 \wedge$ IS-A (A_1, PersonAgeValue)

Show both valid theorems and invalid theorems using UML class diagrams, for both associations and attributes

Formal Semantics of RML/Telos

❖More definitions

❖ PROPVAL (k,, p, s): function which gives the value of the attribute p of element k at time s (the factual property function)

❖ $: a special constant used to denote the null value

❑ necessary (p) $=^{def}$ [PROPDEF (x, p) = y $\wedge$ y $\neq$ $] $\rightarrow$
[IN (z, x, now) $\rightarrow$ PROPVAL (z, p, now) $\neq$ $]

❑ Initially (p) $=^{def}$ [PROPDEF (x, p) = y $\wedge$ y $\neq$ $] $\rightarrow$
[OCCURS (z, x, now, t) $\rightarrow$ PROPDEF (z, p, now) $\neq$ $]

assume that OCCURS has been defined using *meet (later)*

Formal Semantics of RML/Telos

- Basic Predicates – in Set Theory?

  - IN (i, C, s): predicate asserting that token I is an instance of class C

    E.g., IN (John, Child, 1997)
    IN (John, Student, 2004)

  - IS-A (C1, C2): time-independent predicate asserting that class C1 is a subclass of class C2.

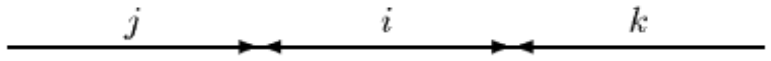
    E.g., IS-A (Child, Person)
    IS-A (Student, Person)

  - PROPDEF (C, a): time-independent function which gives the class to which the value of attribute a for instances of class C must belong (definitional property function)

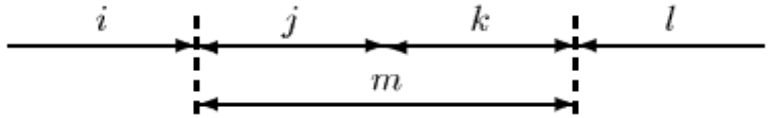
    E.g., PROPDEF (Person, age) => PersonAgeValue
    where Class PersonAgeValue = 0..200
    PROPDEF (Child, age) => ChildAgeValue
    where Class ChildAgeValue = 1..17

RML/Telos – Axiomatization of Temporal Primitives

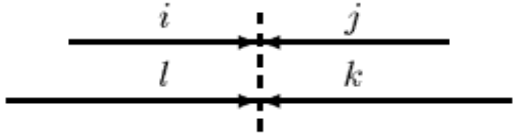
[J. Allen and G. Ferguson, "Actions and Events in Temporal Logic," Journal of Logic and Computation, 1994]

(1) 

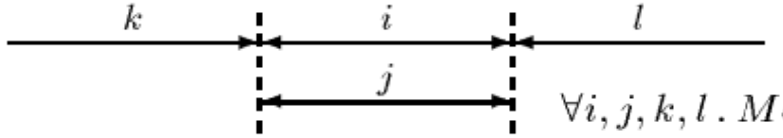
$$\forall i \,.\, \exists j, k \,.\, Meets(j, i) \wedge Meets(i, k).$$

(2) 

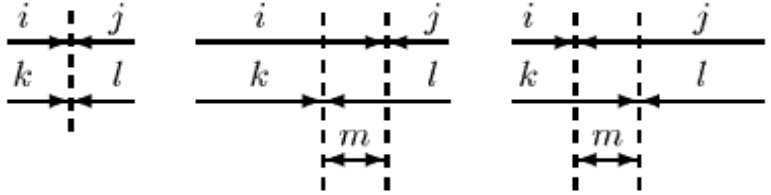
$$\forall i, j, k, l \,.\, Meets(i, j) \wedge Meets(j, k) \wedge Meets(k, l) \supset \exists m \,.\, Meets(i, m) \wedge Meets(m, l).$$

(3) 

$$\forall i, j, k, l \,.\, Meets(i, j) \wedge Meets(i, k) \wedge Meets(l, j) \supset Meets(l, k).$$

(4) 

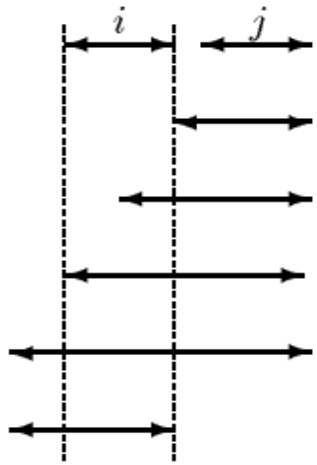
$$\forall i, j, k, l \,.\, Meets(k, i) \wedge Meets(k, j) \wedge Meets(i, l) \wedge Meets(j, l) \supset i = j.$$

(5) 

$$\forall i, j, k, l \,.\, (Meets(i, j) \wedge Meets(k, l)) \supset$$
$$Meets(i, l) \otimes (\exists m \,.\, Meets(k, m) \wedge Meets(m, j)) \otimes$$
$$(\exists m \,.\, Meets(i, m) \vee Meets(m, l)).$$

RML/Telos – Axiomatization of Temporal Primitives

Relation	Inverse
$Before(i,j)$	$After(j,i)$
$Meets(i,j)$	$MetBy(j,i)$
$Overlaps(i,j)$	$OverlappedBy(j,i)$
$Starts(i,j)$	$StartedBy(j,i)$
$During(i,j)$	$Contains(j,i)$
$Finishes(i,j)$	$FinishedBy(j,i)$



$$Before(i,j) \equiv \exists m \,.\, Meets(i,m) \wedge Meets(m,j).$$

$Meets(i,j)$	$i : j$		
$Before(i,j)$	$i \prec j$	$Before(i,j) \vee Meets(i,j)$	$i \prec\!: j$
$During(i,j)$	$i \sqsubset j$	$During(i,j) \vee i = j$	$i \sqsubseteq j$

$Disjoint$: $i \bowtie j \equiv i \prec\!: j \vee j \prec\!: i.$

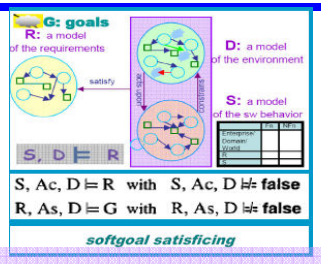
A Brief Survey of Formal Methods

How Much Formality?: How do FMs differ?

Adapted from [http://www.cs.toronto.edu/~sme/CSC2106S/slides/11-how-much-formality.pdf]

- **Mathematical Foundation**
  - **Logic**
    - first order predicate logic - e.g. RML
    - temporal logic - e.g. Albert II, SCR, KAOS
    - multi-valued logic – e.g. Xchek
  - **Other**
    - algebraic languages - e.g. Larch
    - set theory - e.g. Z
- **Ontology**
  - **Fixed**
    - states, events, actions - e.g. SCR
    - entities, activities, assertions - e.g. RML
  - **Extensible**
    - meta language for defining new concepts - e.g. Telos
- **Treatment of Time**
  - **State/event models**
    - time as a discrete sequence of events - e.g. SCR
    - time as quantified intervals - e.g. KAOS
  - **Time as a first class object**
    - meta-level class to represent time - e.g. Telos

Three Different Traditions



[http://www.cs.york.ac.uk/hise/safety-critical-archive/2002/0171.html]

M, Prog |= S; G^s, S, D |= R; (G^s, R, D |= G) V (G^s, R, D |~ G); (G |= ¬P) V (G |~ ¬P)

❑ Formal *Specification* Languages

- Grew out of work on program verification
- Spawned many general purpose specification languages
  - Suitable for specifying the behavior of program units
- Key technologies: Type checking, Theorem proving

Applicability to RE is poor
- No abstraction or structuring
- closely tied to program semantics

Examples: Larch, Z, VDM, …

❑ Reactive System *Modeling*

- Grew out of a need to capture dynamic models of system behavior
- ❑Focus is on reactive systems (e.g. real-time, embedded control systems)
  - support reasoning about safety, liveness
  - ❑ provide a precise requirements specification language
- Key technologies: Consistency checking, Model checking

Applicability to RE is good
- **modeling languages were developed specifically for RE**

Examples: Statecharts, RSML, Parnas-tables, SCR, …

❑ Formal Conceptual *Modeling*

- Grew out of a concern for capturing real-world knowledge in RE
- Focus is on modeling domain entities, activities, agents, assertions
  - provide a formal ontology for domain modeling
  - use first order predicate logic as the underlying formalism
- ❑Key technologies: inference engines, default reasoning, KBS-shells

Applicability to RE is excellent
- **modeling schemes capture key reqs concepts**

Examples: Reqs Apprentice, RML, Telos, The NFR Framework, Albert II, i*

Formal *Specification* Languages

- **Three basic flavors:**
  - **Operational** - specification is executable abstraction of the implementation
    - good for rapid prototyping
    - e.g., Lisp, Prolog, Smalltalk
  - **State-based** - views a program as a (large) data structures whose state can be altered by procedure calls
    - using pre/post-conditions to specify the effect of procedures
    - e.g., VDM, Z
  - **Algebraic** - views a program as a set of abstract data structures with a set of operations…
    - … operations are defined declaratively by giving a set of axioms
    - e.g., Larch, CLEAR, OBJ

- **Developed for specifying *programs***
  - Programs are formal, man-made objects
    - can be modeled precisely in terms of input-output behavior
  - But in RE we're more concerned with:
    - real-world concepts, stakeholders, goals, loosely define problems, environments
  - So these languages are NOT appropriate for RE
    - but people fail to realize that requirements specification ≠ program specification

Reactive System *Modeling*

- **Modeling how a system should behave**
  - Model the environment as a state machine
  - Model the system as a state machine
  - Model safety, liveness properties of the machine as temporal logic assertions
  - Check whether the properties hold of the system interacting with its environment

- **Examples:**
  - Statecharts
    - Harel's notation for modeling large systems
    - Adds parallelism, decomposition and conditional transitions to STDs
  - RSML (Requirements State Machine Language) [Heimdahl & Leveson]
    - Adds tabular specification of complex conditions to Statecharts
  - A7e approach
    - Major project led by Parnas to formalize A7e aircraft requirements spec
    - Uses tables to specify transition relations & outputs
  - SCR (Software Cost Reduction) [Heitmeyer et. al.]
    - Extends the A7e approach to include dictionaries & support tables

Formal Conceptual *Modeling*

- **General approach**
  - model the world beyond functional specifications
    - a specification is prescriptive, concentrating on desired properties of the machine
    - but we also need to capture an understanding of the application domain
    - hence build models of humans' knowledge/beliefs about the world
  - make use of abstraction & refinement as structuring primitives
- **Examples:**
  - RML (Requirements Modeling Language) [Greenspan & Mylopoulos, early-1980s]
    - First major attempt to use knowledge representation techniques in RE
    - Essentially an object oriented language, with classes for activities, entities and assertions
    - Uses First Order Predicate Language as an underlying reasoning engine
  - Telos
    - Extends RML by creating a fully extensible ontology
    - Meta-level classes define the ontology (the basic set is built in)
  - The NFR Framework
    - The first visual modeling of NFRs as softgoals and softgoal satisficing
  - Albert II [Dubois & du Bois, mid-1990s]
    - Models a set of interacting agents that perform actions that change their state
    - Uses an object-oriented real-time temporal logic for reasoning
  - i* [Yu et al]