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 aggregate {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
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]
aggregation/decomposition: categories of attributes
relationships between entities/objects are treated also as objects

Example 2:

<table>
<thead>
<tr>
<th>Book</th>
<th>EntityClass Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>author</td>
<td>author</td>
</tr>
<tr>
<td>firstAuthor</td>
<td>firstAuthor: Person</td>
</tr>
<tr>
<td>secondAuthor</td>
<td>secondAuthor: Person</td>
</tr>
<tr>
<td>title</td>
<td>title: string</td>
</tr>
</tbody>
</table>

generalization/specialization: infinite hierarchy of classes
tokens, classes, metaclasses, metametaclasses, ... omegaclass

Recall:
an object is a "concept" of anything (including 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

\[ M = \{ A \mid A \not\in A \}. \]

is M an element of itself?

[A. Church, Theory of Types, 1940.]
Uniform treatment of attributes and associations
Activities

* SADT:

\[ I_1 \rightarrow \text{Persons} \rightarrow \text{Patient} \rightarrow \text{Patients} \rightarrow O_1 \]

\[ \text{MD} \rightarrow \text{Nursing-Homes} \]

\[ \text{C1} \quad \text{C2} \]

\[ ? \rightarrow \text{Employee} \]

* OO-RML

ActivityClass AdmitPatients with

- input
  - person: Persons
- control
  - toHome: Nursing-Homes
  - doc: MD
- mechanism
  - clerk: Employee
- output
  - 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)

; activation condition, termination condition
; when should an activity start and end
; initially (preconditions) and finally (postconditions):
; what conditions should hold for an activity to start;
; what conditions should hold at the time of termination

Lawrence Chung
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
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

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

*see the axiomatization later on
A Formalization of RML/Telos

[S. Greenspan, J. Mylopoulos and A. Borgida, A. “A Requirements Modeling Language and Its Logic,”
Formal Semantics

- **formal = syntax = grammar**
  - reasoning, consistency checking, derive new facts
  - deductive reasoning
  - abductive reasoning
  - approximate reasoning
  - analogical reasoning
  - qualitative reasoning
  - probabilistic reasoning
  - ...

- **semantics: meaning**
  - 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

- **connectives:**
  - ^, & logical conjunction
  - v 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

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

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

---

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 \( \mathcal{P} \), written \( A_1, \ldots, A_n \vdash_{\mathcal{P}} B \) if there is a derivation for \( B \) in \( \mathcal{P} \) with premises \( A_1, \ldots, A_n \).

  *Every derivable formula is valid.*

- \( \vdash_{\mathcal{P}} \Lambda(A) \iff \vdash_{\mathcal{P}} \Lambda(B) \)

  *Every valid formula is derivable.*


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

Partial Order IS-A Constraint

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

- anti-symmetric: \((C_1 \neq C_2) \land IS-A (C_1, C_2) \leftrightarrow \neg IS-A (C_2, C_1)\)
  - E.g., IS-A (Child, Person) \(\rightarrow\) \(\neg IS-A (Child, Person)\)

- transitive: IS-A (\(C_1, C_2\)) \(\land\) IS-A (\(C_2, C_3\)) \(\rightarrow\) IS-A (\(C_1, C_3\))
  - E.g., IS-A (ChildStudent, Child) \(\land\) 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 (\(C_2, C_3\)) \(\land\) IS-A (\(C_1, C_2\)) \(\land\) IS-A (\(C_3, C_1\))
Formal Semantics of RML/Telos

Axioms

Extensional IS-A Constraint

\[ \text{IN} (I, C_1, s) \land \text{IS-A} (C_1, C_2) \rightarrow \text{IN} (I, C_2, s) \]

Each instance of a class is in its superclass

- \( \text{IN} (\text{Cathy}, \text{Child}, 2004) \land \text{IS-A} (\text{Child}, \text{Person}) \rightarrow \text{IN} (\text{cathy}, \text{Person}, 2004)? \)
- \( \text{IN} (\text{Maria}, \text{Person}, 2004) \land \text{IS-A} (\text{Child}, \text{Person}) \rightarrow \text{IN} (\text{Maria}, \text{Child}, 2004)? \)
- \( \text{IN} (\text{Chris}, \text{Child}, 2004) \land \text{IS-A} (\text{Child}, \text{Person}) \rightarrow \text{IN} (\text{Chris}, \text{Person}, 2005)? \)
- \( \text{IN} (\text{Person}, \text{PersonClass}, 1996) \land \text{IN} (\text{John}, \text{Person}, 2004) \rightarrow \text{IN} (\text{John}, \text{PersonClass}, 2004)? \)
Formal Semantics of RML/Telos

- Axioms

- Intensional IS-A Constraint

\[ \text{PROPDEF} \( (C_2, a) = A_2 \land \text{IS-A} \( C_1, C_2) \) \]
\[ \exists A_1 \ [ \text{PROPDEF} \( C_1, a) = A_1 \land \text{IS-A} \( A_1, A_2) \)] \]

Inherited attributes have more specialized ranges

-\[ \text{PROPDEF} \( \text{Person}, \text{age}) = \text{PersonAge Value} \land \text{IS-A} \( \text{Child}, \text{Person}) \]
\[ \exists A_1 \ [ \text{PROPDEF} \( \text{Child}, \text{age}) = A_1 \land \text{IS-A} \( A_1, \text{PersonAgeValue}) \]

Show both valid theorems and invalid theorems using UML class diagrams, for both associations and attributes
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) =\[\text{def } [\text{PROPDEF } (x, p) = y \land y \neq $] \rightarrow \]
\[\text{IN } (z, x, \text{now}) \rightarrow \text{PROPVAL } (z, p, \text{now}) \neq $]

Initially (p) =\[\text{def } [\text{PROPDEF } (x, p) = y \land y \neq $] \rightarrow \]
\[\text{OCCURS } (z, x, \text{now}, t) \rightarrow \text{PROPDEF } (z, p, \text{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
PROPDEF (Child, age) => ChildAgeValue

where Class PersonAgeValue = 0..200
where Class ChildAgeValue = 0..17
RML/Telos – Axiomatization of Temporal Primitives


(1) \(j \quad i \quad k\) \quad \forall i . \exists j, k . \text{Meets}(j, i) \land \text{Meets}(i, k).

(2) \(i \quad j \quad k \quad l\) \quad \exists m . \text{Meets}(i, m) \land \text{Meets}(m, l).

(3) \(i \quad j \quad k \quad l\) \quad \forall i, j, k, l . \text{Meets}(i, j) \land \text{Meets}(j, k) \land \text{Meets}(k, l) \supset \exists m . \text{Meets}(i, m) \land \text{Meets}(m, l).

(4) \(i \quad j \quad l\) \quad \forall i, j, k, l . \text{Meets}(k, i) \land \text{Meets}(k, j) \land \text{Meets}(i, l) \land \text{Meets}(j, l) \supset i = j.

(5) \(i \quad j \quad k \quad l\) \quad \forall i, j, k, l . (\text{Meets}(i, j) \land \text{Meets}(k, l)) \supset \text{Meets}(i, l) \otimes (\exists m . \text{Meets}(k, m) \land \text{Meets}(m, j)) \otimes (\exists m . \text{Meets}(i, m) \lor \text{Meets}(m, l)).
# RML/Telos – Axiomatization of Temporal Primitives

<table>
<thead>
<tr>
<th>Relation</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before ((i, j))</td>
<td>After ((j, i)) (\land) (\exists m.) Meets ((i, m)) (\land) Meets ((m, j)).</td>
</tr>
<tr>
<td>Meets ((i, j))</td>
<td>MetBy ((j, i))</td>
</tr>
<tr>
<td>Overlaps ((i, j))</td>
<td>OverlappedBy ((j, i))</td>
</tr>
<tr>
<td>Starts ((i, j))</td>
<td>StartedBy ((j, i))</td>
</tr>
<tr>
<td>During ((i, j))</td>
<td>Contains ((j, i))</td>
</tr>
<tr>
<td>Finishes ((i, j))</td>
<td>FinishedBy ((j, i))</td>
</tr>
</tbody>
</table>

\(\text{Meets}(i, j) \leftrightarrow i : j\)

\(\text{Before}(i, j) \leftrightarrow i \prec j\)

\(\text{During}(i, j) \leftrightarrow i \sqsubset j\)

\(\text{Disjoint}: i \bowtie j \equiv i \preceq j \lor j \preceq i.\)
A Brief Survey of Formal Methods

How Much Formality?: How do FMs differ?


- **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

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]