Dynamic Logic Semantics for UML Consistency

Greg O'Keefe

Computer Sciences Laboratory
Australian National University

The Big Picture

a model is a statement about systems

The Big Picture

a model is a statement about systems

- but its meaning is only approximate

The Big Picture

a model is a statement about systems

- but its meaning is only approximate
- so we can not say if it is consistent

The Big Picture

a model is an approximate statement about systems

The Big Picture

a model is an approximate statement about systems

a dynamic logic formula is also a statement about systems

The Big Picture

a model is an approximate statement about systems

a dynamic logic formula is also a statement about systems

- its meaning is exact

The Big Picture

a model is an approximate statement about systems

a dynamic logic formula is also a statement about systems

- its meaning is exact
- and we can automatically determine its consistency

The Big Picture

a model is an approximate statement about systems

a dynamic logic formula is also a statement about systems

- its meaning is exact
- and we can automatically determine its consistency (undecidable, but most of the time)

The Big Picture

a model is an approximate statement about systems

a dynamic logic formula is a precise statement about systems

The Big Picture

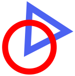
a model is an approximate statement about systems

a dynamic logic formula is a precise statement about systems

so we translate models into dynamic logic ...

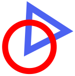
Consistency (Logic 101)

In a situation, each statement is true or false.

	The circle is red.	*true*
	The circle is red.	*false*

Consistency (Logic 101)

In a situation, each statement is true or false.



The circle is red. *true*

The circle is red. *false*

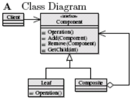
A statement is *consistent* if it is true in some situation.

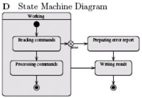
The circle is red.	*consistent*
The circle is square.	*inconsistent*

A model is a statement about systems

Given a system, each model is true or false.

situation/system | statement/model



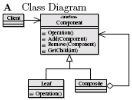


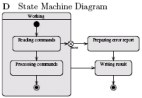
true? false?

A model is a statement about systems

Given a system, each model is true or false.

situation/system | statement/model





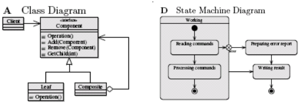
true? false?

To answer model consistency questions, we need definitions of:

A model is a statement about systems

Given a system, each model is true or false.

situation/system | statement/model



true? false?

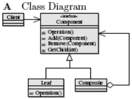
To answer model consistency questions, we need definitions of:

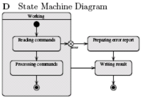
- model (syntax)
- system (semantic domain)

A model is a statement about systems

Given a system, each model is true or false.

situation/system　　　statement/model





true? false?

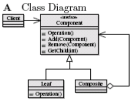
To answer model consistency questions, we need definitions of:

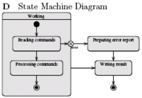
- model (syntax)
- system (semantic domain)
- when a model is true of a system (semantics)

A model is a statement about systems

Given a system, each model is true or false.

situation/system | statement/model





true? false?

To answer model consistency questions, we need definitions of:

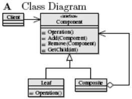
- model (syntax)
- system (semantic domain)
- when a model is true of a system (semantics)

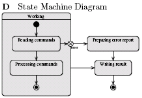
Do we have this in the OMG documents?

A model is a statement about systems

Given a system, each model is true or false.

situation/system statement/model





true? false?

To answer model consistency questions, we need definitions of:

- model (syntax)
- system (semantic domain)
- when a model is true of a system (semantics)

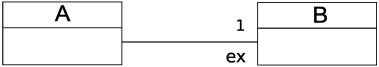
Do we have this in the OMG documents?

To answer the questions automatically, we need

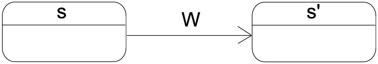
a procedure to search the (infinte) space of systems.

Mellor's Challenge

Class Diagram



State Machine for Class *A*

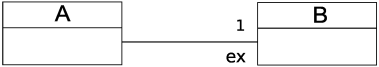


Entry Action for State s'

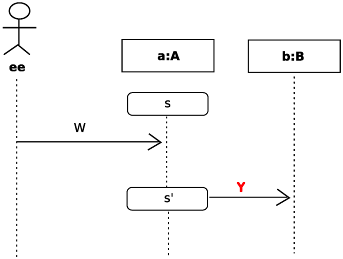
```
send X to self.ex
```

Mellor's Challenge

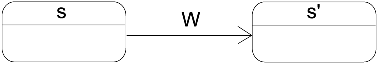
Class Diagram



Sequence Diagram



State Machine for Class *A*



Entry Action for State s'

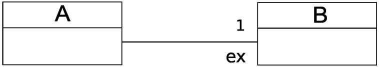
`send X to self.ex`

Mellor's Challenge

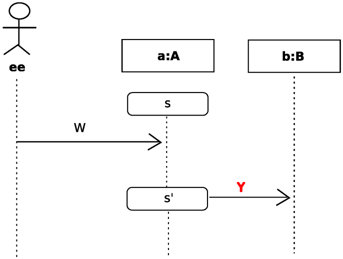
We want

- semantics to say this is inconsistent

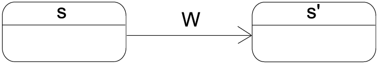
Class Diagram



Sequence Diagram



State Machine for Class *A*



Entry Action for State s'

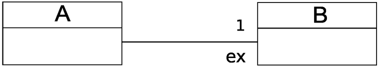
```
send X to self.ex
```

Mellor's Challenge

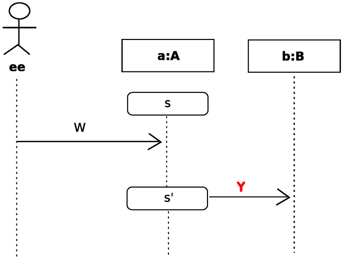
We want

- semantics to say this is inconsistent
- tools to detect it

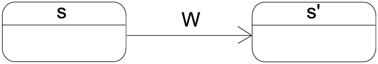
Class Diagram



Sequence Diagram



State Machine for Class *A*



Entry Action for State s'

```
send X to self.ex
```

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic

syntax example: $\langle y := f(x) \rangle x = y$

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic

syntax example: $\langle y := f(x) \rangle x = y$

- $\langle \texttt{program} \rangle \varphi$ means φ might be true after `program` runs

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic

syntax example: $\langle y := f(x) \rangle x = y$

- $\langle \texttt{program} \rangle \varphi$ means φ might be true after `program` runs
- `program` means binary relation over valuations

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic

syntax example: $\langle y := f(x) \rangle x = y$

- $\langle \texttt{program} \rangle \varphi$ means φ might be true after `program` runs
- `program` means binary relation over valuations
- $x := t$ relates u to the x-variant with $x \mapsto t^{\mathfrak{M},u}$

Dynamic Logic (Logic 201)

First Order Logic

syntax example: $\forall x \bullet f(x) = y$

- interpretation $\mathfrak{M}$ gives us a function $f^{\mathfrak{M}}$
- valuation u gives us individuals x^u, y^u
- $\forall x$ needs truth of $f(x) = y$ under all x-variants of u
- example formula is true iff $f^{\mathfrak{M}}$ is constant with value y^u

Dynamic Logic

syntax example: $\langle y := f(x) \rangle x = y$

- $\langle \texttt{program} \rangle \varphi$ means φ might be true after `program` runs
- `program` means binary relation over valuations
- $x := t$ relates u to the x-variant with $x \mapsto t^{\mathfrak{M},u}$
- more syntax: $\rho ; \rho'$ $\rho \cup \rho'$ ρ^* $\varphi ?$ $[\rho]\varphi$

System States and Evolution

Statics: What is a system state?

System States and Evolution

Statics: What is a system state?

- a system state is a valuation

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist
- attributes, association ends are “array” variables

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist
- attributes, association ends are “array” variables

Dynamics: How can a system evolve?

Objects do actions, if conditions allow:

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist
- attributes, association ends are “array” variables

Dynamics: How can a system evolve?

Objects do actions, if conditions allow: `guard`?; `action`

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist
- attributes, association ends are “array” variables

Dynamics: How can a system evolve?

Objects do actions, if conditions allow:

$$\varepsilon \equiv ((sc(x, M, y)?;\ x.\texttt{send}\ M\ \texttt{to}\ y) \cup (ac(x)?;\ x.\texttt{accept}))^*$$

System States and Evolution

Statics: What is a system state?

- a system state is a valuation
- objects are individuals, they persist
- attributes, association ends are “array” variables

Dynamics: How can a system evolve?

Objects do actions, if conditions allow:

$$\varepsilon \equiv ((sc(x, M, y)?;\ x.\mathtt{send}\ M\ \mathtt{to}\ y) \cup (ac(x)?;\ x.\mathtt{accept}))^*$$

$$\begin{aligned} sc(x, M, y) &\equiv x.class = ExternalEntity \\ &\vee (head(x.todo) = \mathtt{send}\ M\ \mathtt{to}\ y) \\ x.\mathtt{send}\ M\ \mathtt{to}\ y &\equiv y.intray := append(y.intray, M); \\ & \quad x.todo := tail(x.todo) \end{aligned}$$

Class Diagram

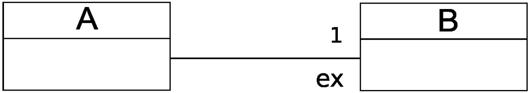
For each diagram, a range of interpretations is possible, even desirable. Here we give rather weak ones.

Class Diagram

For each diagram, a range of interpretations is possible, even desirable. Here we give rather weak ones. *(They are shorter!)*

Class Diagram

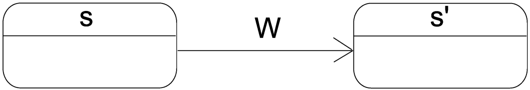
For each diagram, a range of interpretations is possible, even desirable. Here we give rather weak ones. *(They are shorter!)*



$$CD \equiv [\varepsilon](\forall x \bullet x.class = A \longrightarrow size(x.ex) = 1 \wedge (\forall y \bullet y \in x.ex \longrightarrow y.class = B))$$

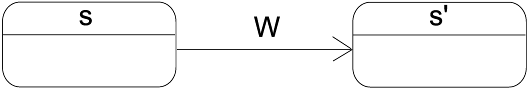
State Machine Diagram

We do not yet specify which objects the state machine diagram applies to, so the formulae have a free variable.



State Machine Diagram

We do not yet specify which objects the state machine diagram applies to, so the formulae have a free variable.



$$SM_s(x) \equiv [\varepsilon](x.state = s \lor x.state = s')$$

$$SM_t(x) \equiv [\varepsilon](x.state = s \land head(x.intray) = W \longrightarrow [x.\texttt{accept}]\, x.state = s')$$

Weaving as Formation

Aspect Oriented Modelling and model “weaving” are hot research topics. In this formal setting, it is clear and simple.

Weaving as Formation

Aspect Oriented Modelling and model “weaving” are hot research topics. In this formal setting, it is clear and simple.

action - state join

Put action on *todo* list when object enters state.

$$\begin{aligned} SM_p(x) \equiv \quad & [\varepsilon][x.\mathtt{accept}](\\ & x.state = s' \longrightarrow \\ & x.todo = \mathtt{send}\ X\ \mathtt{to}\ x.\mathtt{ex}) \end{aligned}$$

Weaving as Formation

Aspect Oriented Modelling and model “weaving” are hot research topics. In this formal setting, it is clear and simple.

action - state join

Put action on *todo* list when object enters state.

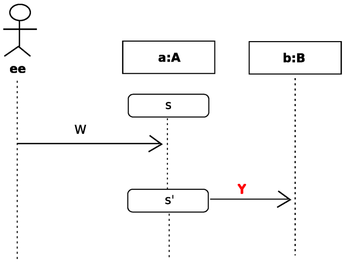
$$SM_p(x) \equiv \; [\varepsilon][x.\mathtt{accept}](\\ x.state = s' \longrightarrow \\ x.todo = \mathtt{send}\ X\ \mathtt{to}\ x.\mathtt{ex})$$

state machine - class join

Make objects of class A obey state machine formulae.

$$SM \equiv [\varepsilon](\forall x \bullet x.class = A \longrightarrow SM_s(x) \wedge SM_t(x) \wedge SM_p(x))$$

Sequence Diagram



$$
\begin{aligned}
SEQ \equiv \quad & class(ee) = ExternalEntity \wedge \\
& a.class = A \ \wedge \ b.class = B \wedge \\
& \langle \varepsilon \rangle (sc(ee, W, a) \wedge \langle ee.\mathtt{send}\ W\ \mathtt{to}\ a \rangle \\
& \langle \varepsilon \rangle (ac(a) \wedge \langle a.\mathtt{accept} \rangle \\
& \langle \varepsilon \rangle (sc(a, Y, b) \wedge \langle a.\mathtt{send}\ Y\ \mathtt{to}\ b \rangle \top)))
\end{aligned}
$$

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Our search

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Our search

- we drop $CD \wedge SM \wedge SEQ$ into a tableau prover, turn the handle and then . . .

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Our search

- we drop $CD \wedge SM \wedge SEQ$ into a tableau prover, turn the handle and then ...
- it gives us a system where $X = Y$, showing that the UML model *is* consistent, hmmm!

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Our search

- we drop $CD \wedge SM \wedge SEQ$ into a tableau prover, turn the handle and then . . .
- it gives us a system where $X = Y$, showing that the UML model *is* consistent, hmmm!
- so next time we add $X.name =$ "X" etc. to our theory

Consistency: the search for a satisfying system

Semantic tableaux theorem provers

- a formula φ is valid iff $\neg\varphi$ is inconsistent
- if a complete search for an interpretation to satisfy $\neg\varphi$ finds none, then it is a proof of φ
- we can use these interpretation finders to demonstrate model consistency

Our search

- we drop $CD \wedge SM \wedge SEQ$ into a tableau prover, turn the handle and then . . .
- it gives us a system where $X = Y$, showing that the UML model *is* consistent, hmmm!
- so next time we add $X.name =$ “X” etc. to our theory
- and then the UML model can be shown inconsistent

Summary

- By translating models into dynamic logic we

Summary

- By translating models into dynamic logic we
  - give precise meaning

Summary

- By translating models into dynamic logic we
  - give precise meaning
  - enable consistency check

Summary

- By translating models into dynamic logic we
  - give precise meaning
  - enable consistency check
- Why DL? Why not TLA+, Z, ASM's, OCL (?!), ...?

Summary

- By translating models into dynamic logic we
  - give precise meaning
  - enable consistency check
- Why DL? Why not TLA+, Z, ASM’s, OCL (?!), ...?
- With DL we have made *action outline statements*.

Summary

- By translating models into dynamic logic we
  - give precise meaning
  - enable consistency check
- Why DL? Why not TLA+, Z, ASM’s, OCL (?!), ...?
- With DL we have made *action outline statements*.
  - ignore irrelevant detail

Summary

- By translating models into dynamic logic we
  - give precise meaning
  - enable consistency check
- Why DL? Why not TLA+, Z, ASM's, OCL (?!), ...?
- With DL we have made *action outline statements*.
  - ignore irrelevant detail
  - raise the level of abstraction