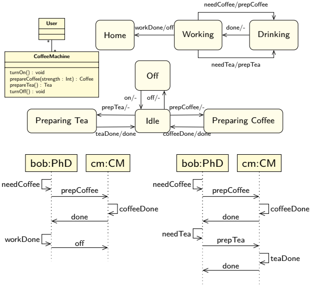
Guided Merging of Sequence Diagrams

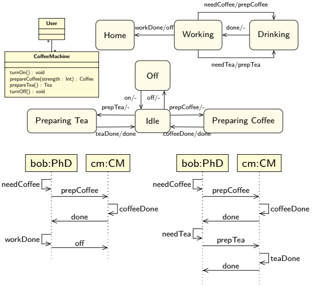
Magdalena Widl

Knowledge-Based Systems Group
Vienna University of Technology

Motivation and Scope

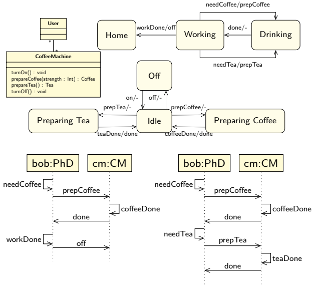


Motivation and Scope



Model-Driven Engineering

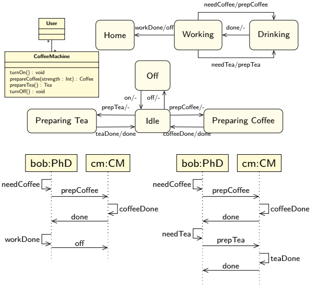
Motivation and Scope



Model-Driven Engineering

Model Evolution

Motivation and Scope

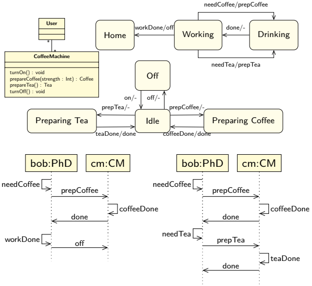


Model-Driven Engineering

Model Evolution

Model Versioning

Motivation and Scope



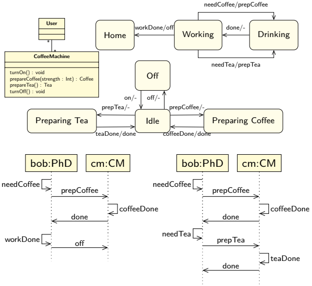
Model-Driven Engineering

Model Evolution

Model Versioning

Model Merging

Motivation and Scope



Model-Driven Engineering

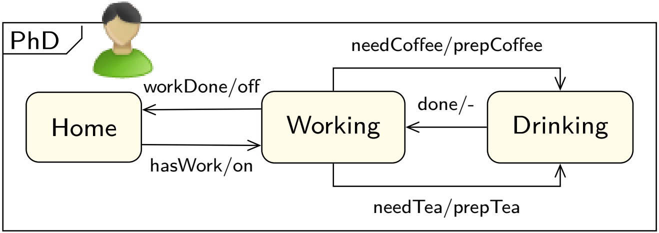
Model Evolution

Model Versioning

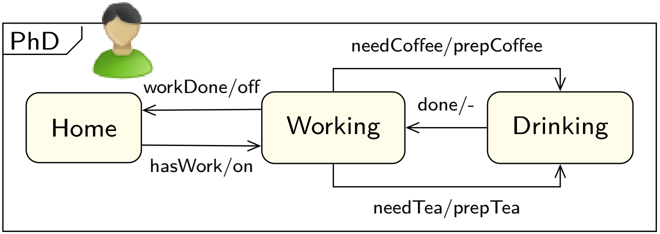
Model Merging

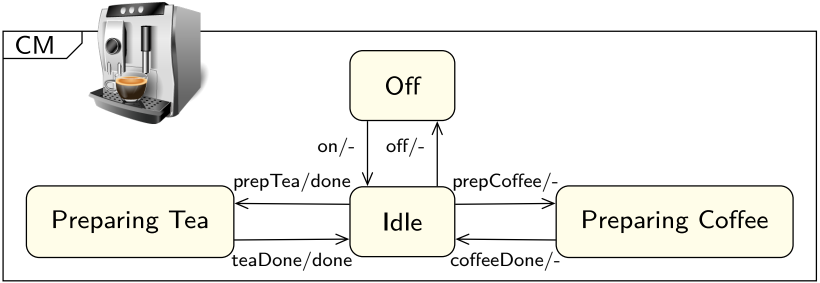
Multi-View Modelling

Example

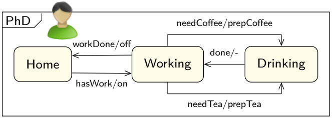

PhD
needCoffee/prepCoffee
workDone/off
done/-
Home
Working
Drinking
hasWork/on
needTea/prepTea

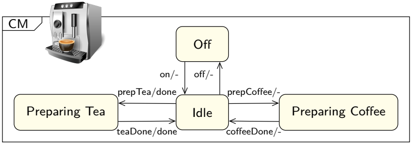
Example


PhD
needCoffee/prepCoffee
workDone/off
Home
Working
done/-
Drinking
hasWork/on
needTea/prepTea

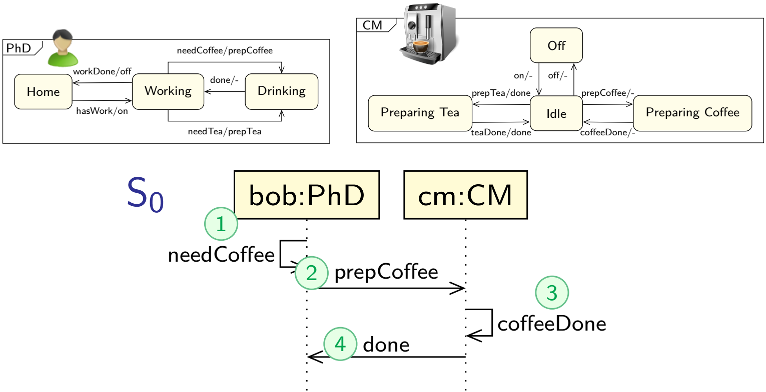

CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-

Example

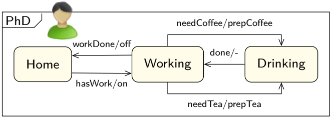

PhD
Home
Working
Drinking
workDone/off
hasWork/on
needCoffee/prepCoffee
done/-
needTea/prepTea

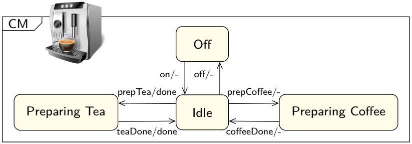

CM
Off
on/-
off/-
Idle
Preparing Tea
Preparing Coffee
prepTea/done
teaDone/done
prepCoffee/-
coffeeDone/-

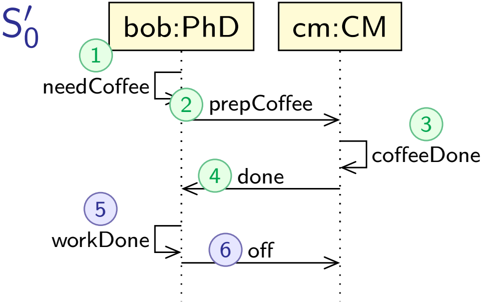
Example


PhD
needCoffee/prepCoffee
workDone/off
Home
Working
done/-
Drinking
hasWork/on
needTea/prepTea
CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-
S_0
bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done

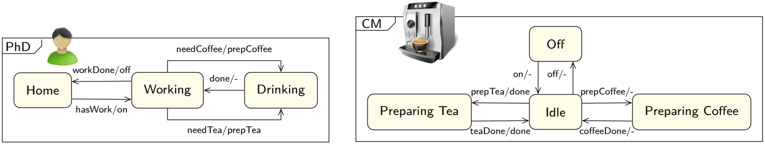
Example


PhD
needCoffee/prepCoffee
workDone/off
done/-
Home
Working
Drinking
hasWork/on
needTea/prepTea

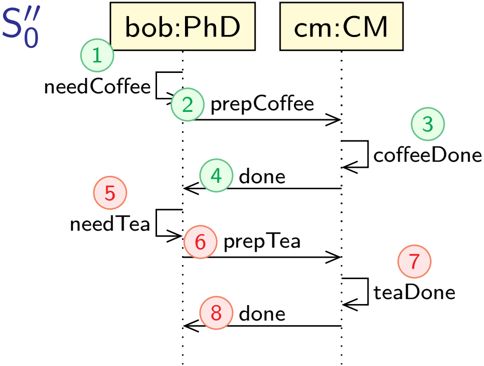

CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-


S_0'
bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
workDone
6
off

Example

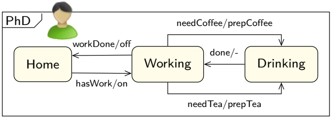


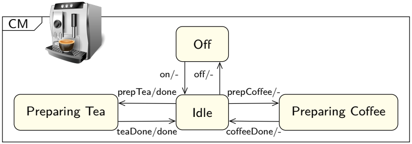
PhD
Home
Working
Drinking
workDone/off
hasWork/on
done/-
needCoffee/prepCoffee
needTea/prepTea
CM
Off
Idle
Preparing Tea
Preparing Coffee
on/-
off/-
prepTea/done
teaDone/done
prepCoffee/-
coffeeDone/-

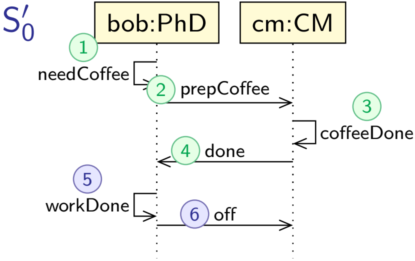


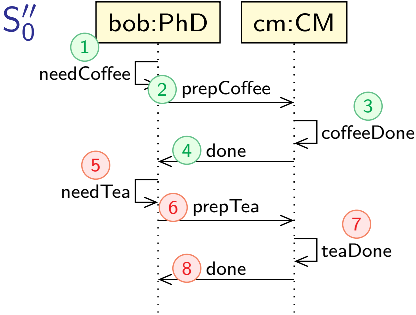
S_0''
bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
needTea
6
prepTea
7
teaDone
8
done

Example

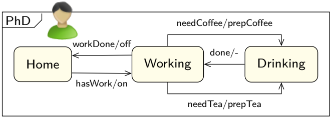

PhD
needCoffee/prepCoffee
workDone/off
done/-
Home
Working
Drinking
hasWork/on
needTea/prepTea

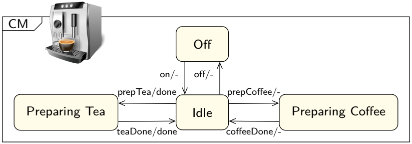

CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-


S'_0
bob:PhD
cm:CM
1 needCoffee
2 prepCoffee
3 coffeeDone
4 done
5 workDone
6 off

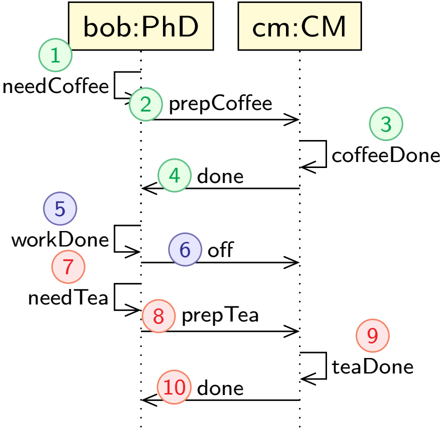

S''_0
bob:PhD
cm:CM
1 needCoffee
2 prepCoffee
3 coffeeDone
4 done
5 needTea
6 prepTea
7 teaDone
8 done

Example

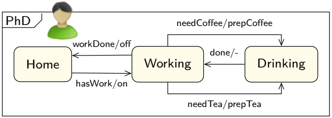

PhD
Home
Working
Drinking
workDone/off
hasWork/on
done/-
needCoffee/prepCoffee
needTea/prepTea

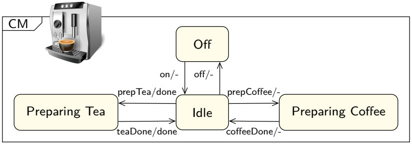

CM
Off
Idle
Preparing Tea
Preparing Coffee
on/-
off/-
prepTea/done
teaDone/done
prepCoffee/-
coffeeDone/-

S_1 ?

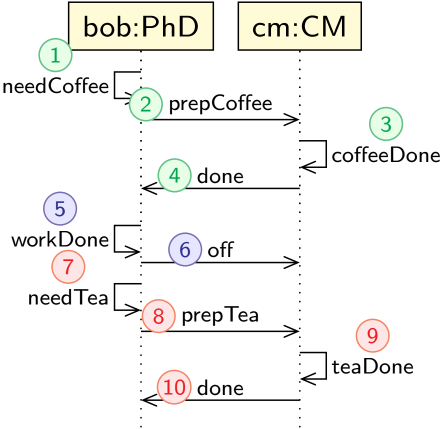

bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
workDone
6
off
7
needTea
8
prepTea
9
teaDone
10
done

Example

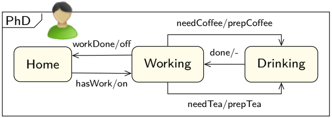

PhD
Home
Working
Drinking
workDone/off
hasWork/on
done/-
needCoffee/prepCoffee
needTea/prepTea


CM
Off
Idle
Preparing Tea
Preparing Coffee
on/-
off/-
prepTea/done
teaDone/done
prepCoffee/-
coffeeDone/-

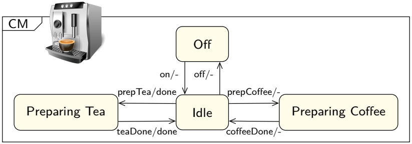
S_1 ?


bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
workDone
6
off
7
needTea
8
prepTea
9
teaDone
10
done

Example

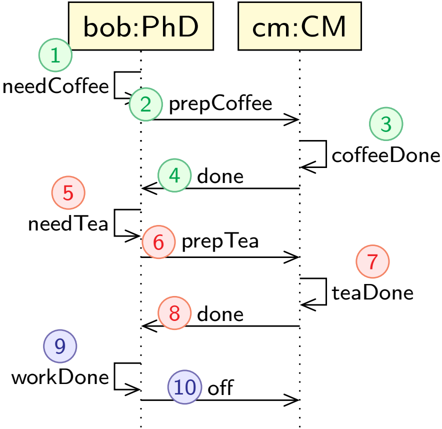


PhD
needCoffee/prepCoffee
workDone/off
done/-
Home
Working
Drinking
hasWork/on
needTea/prepTea



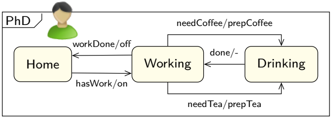
CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-

S_1 ?

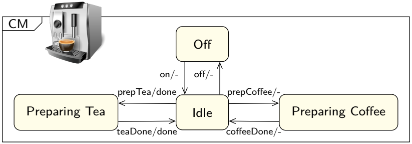


bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
needTea
6
prepTea
7
teaDone
8
done
9
workDone
10
off

Example

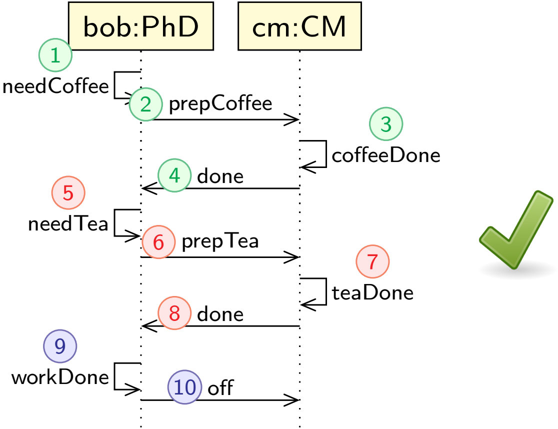


PhD
needCoffee/prepCoffee
workDone/off
done/-
Home
Working
Drinking
hasWork/on
needTea/prepTea



CM
Off
on/-
off/-
prepTea/done
prepCoffee/-
Preparing Tea
Idle
Preparing Coffee
teaDone/done
coffeeDone/-

S_1 ?



bob:PhD
cm:CM
1
needCoffee
2
prepCoffee
3
coffeeDone
4
done
5
needTea
6
prepTea
7
teaDone
8
done
9
workDone
10
off

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations
- A scalable algorithm that solves the problem

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations
- A scalable algorithm that solves the problem

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations
- A scalable algorithm that solves the problem

Our Contributions

- Formalization of state machine, sequence diagram, formal problem statement

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations
- A scalable algorithm that solves the problem

Our Contributions

- Formalization of state machine, sequence diagram, formal problem statement
- Translation to propositional SAT Problem

Guided Merging of Sequence Diagrams

Why? No more cumbersome manual merging!

Challenges

- Model merging is much more complex than text merging
- Formal semantics for inter-diagram relations
- A scalable algorithm that solves the problem

Our Contributions

- Formalization of state machine, sequence diagram, formal problem statement
- Translation to propositional SAT Problem
- Implementation and evaluation

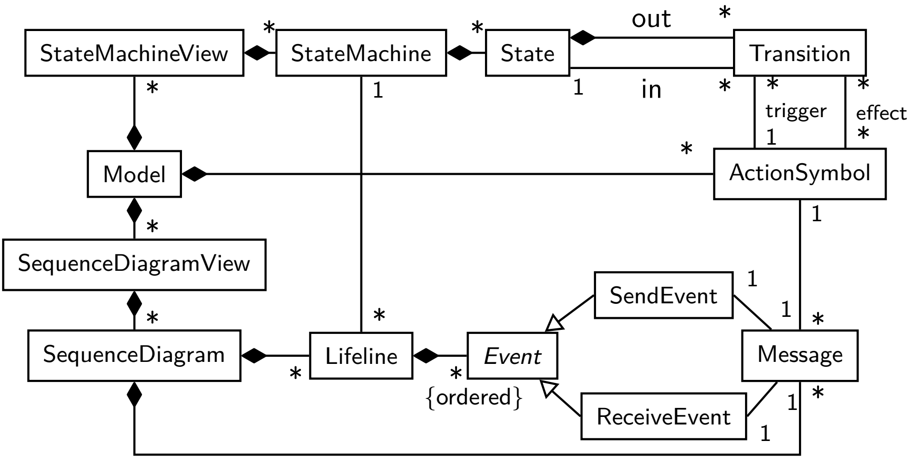
Guided Merging of Sequence Diagrams

Our Contributions

- **Formalization of state machine, sequence diagram, formal problem statement**
- Translation to propositional SAT Problem
- Implementation and evaluation

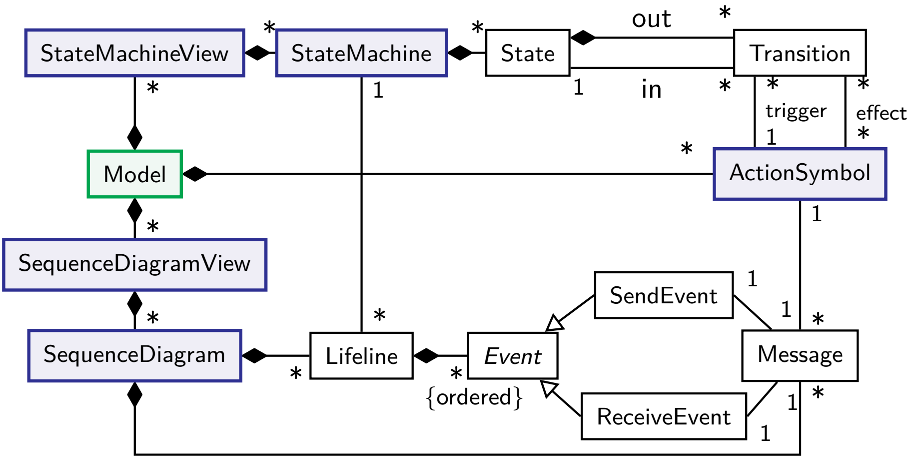
Formalization

The *tMVML* Metamodel



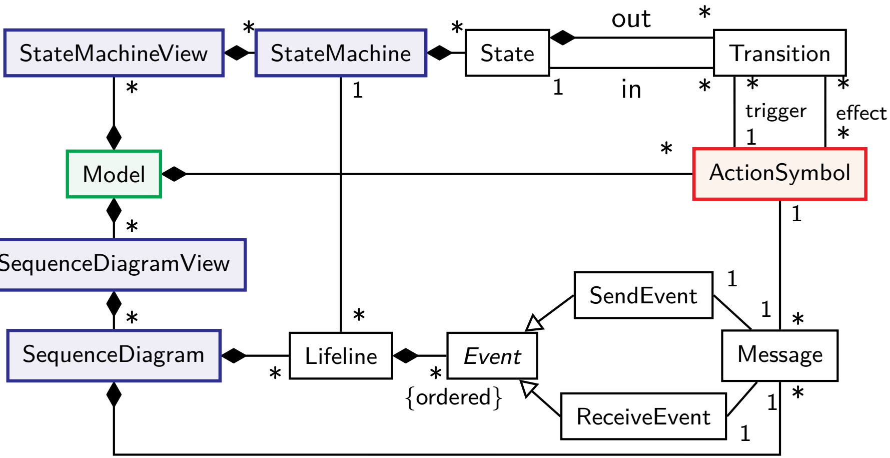
Formalization

The *tMVML* Metamodel



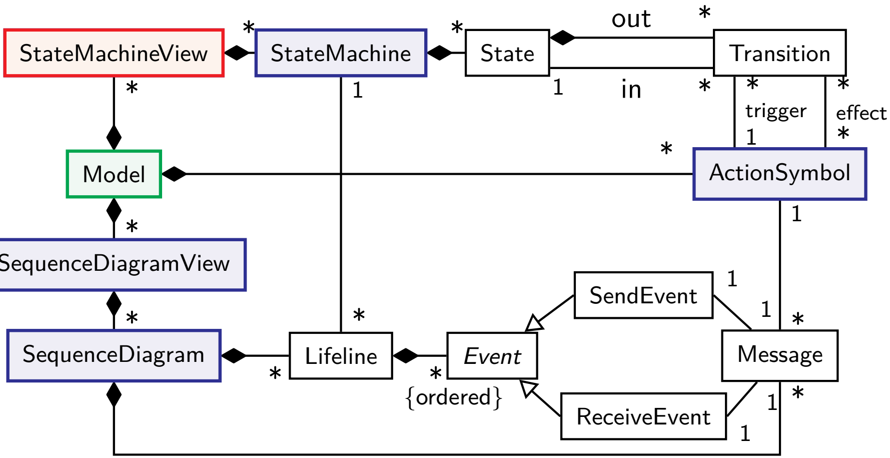
Formalization

The *tMVML* Metamodel



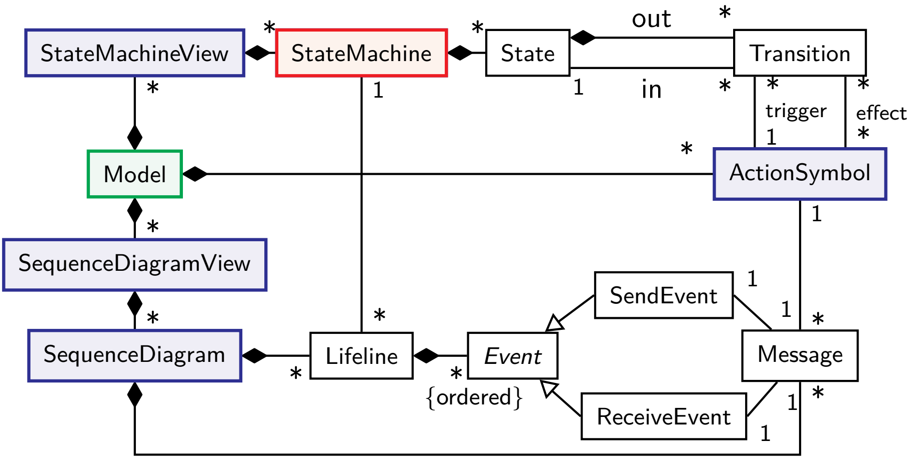
Formalization

The *tMVML* Metamodel



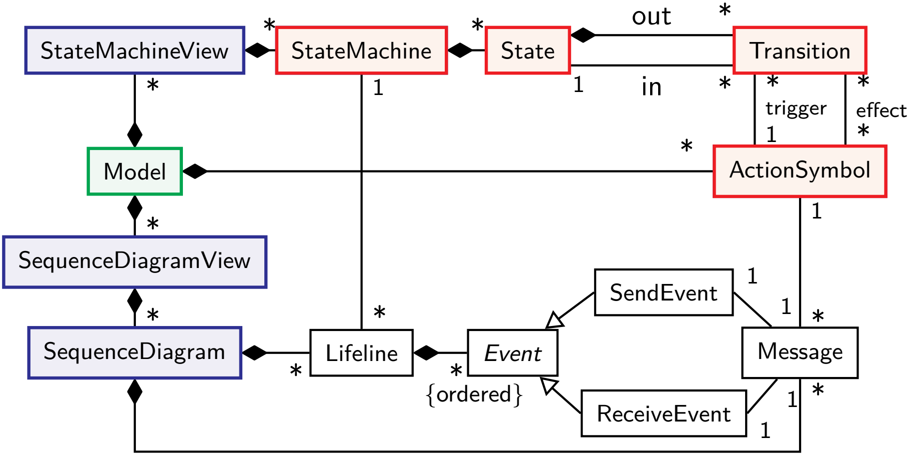
Formalization

The *tMVML* Metamodel



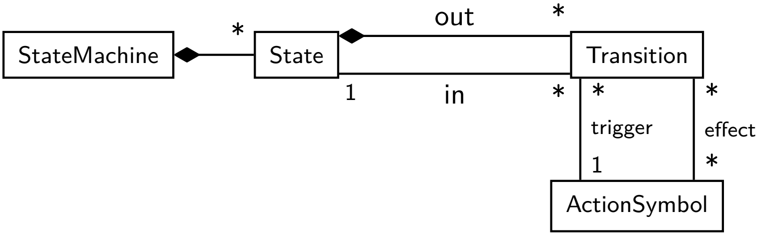
Formalization

The *tMVML* Metamodel



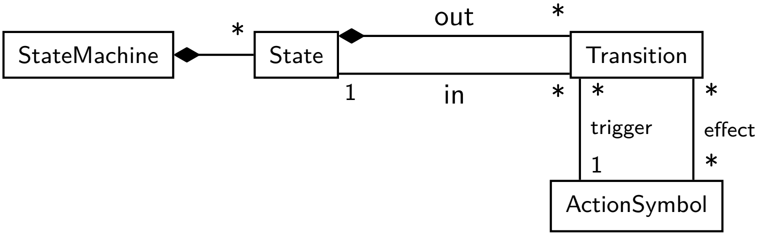
Formalization

State Machine



Formalization

State Machine

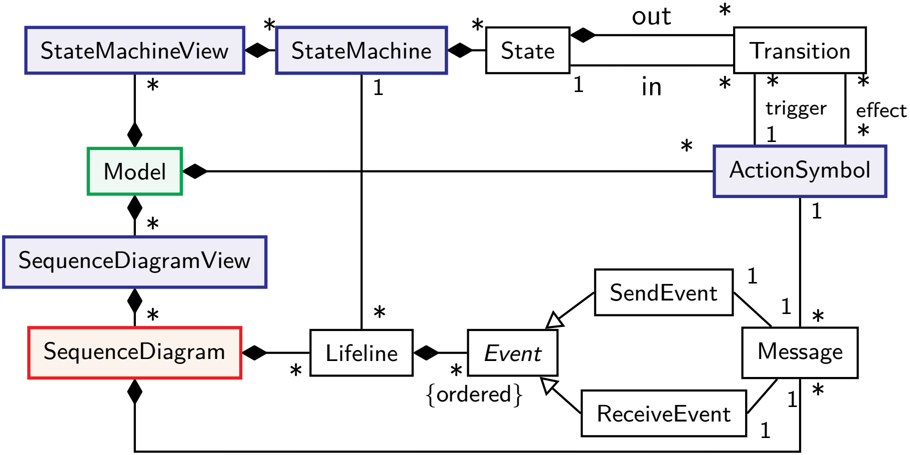


Given an alphabet $\mathcal{A}_A$, a state machine is a quadruple (S, A^{tr}, A^{eff}, T), where

- S is a set of states,
- $A^{tr}, A^{eff} \subseteq \mathcal{A}_A$ are sets of action symbols, and
- $T \subseteq (S \times A^{tr} \times \mathcal{P}(A^{eff}) \times S)$ is a relation representing the transitions between states.

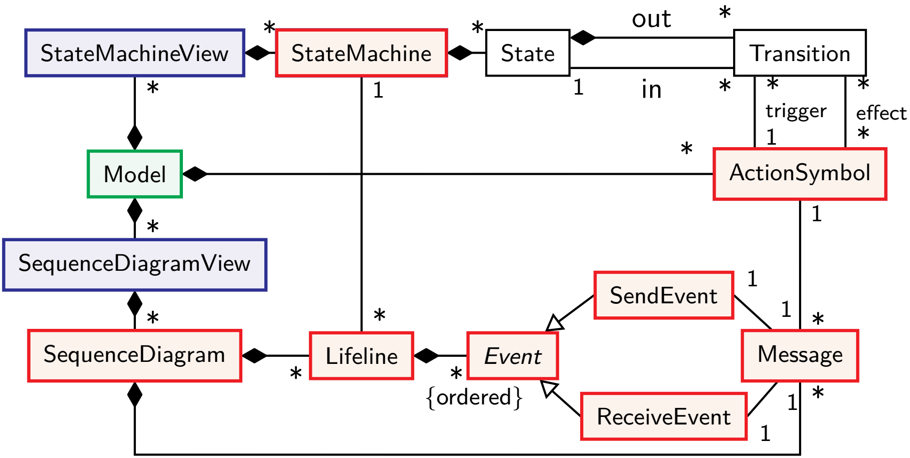
Formalization

The *tMVML* Metamodel



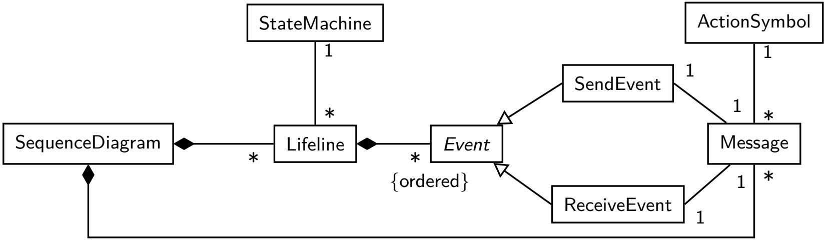
Formalization

The *tMVML* Metamodel



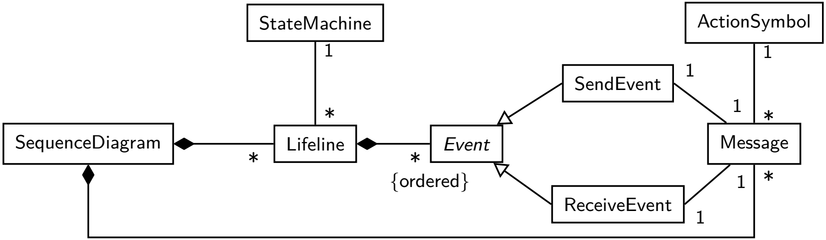
Formalization

Sequence Diagram



Formalization

Sequence Diagram



Given the alphabets $\mathcal{A}_A$ and $\mathcal{A}_E$, and a set $\mathcal{SM}$ of state machines, a sequence diagram is a quadruple $(L, M, \mathsf{life}, \mathsf{msg})$, where

- L is a set of lifelines,
- M is a set of messages,
- life : $L \to (\mathcal{SM} \times \mathcal{P}(\mathcal{A}_E) \times \mathcal{P}(\mathcal{A}_E) \times \mathcal{P}(\mathcal{A}_E \times \mathcal{A}_E))$
- msg : $M \to (\mathcal{A}_A \times \bigcup_{l \in L} \pi_2(\mathsf{life}(l)) \times \bigcup_{l \in L} \pi_3(\mathsf{life}(l)))$.

Formalization

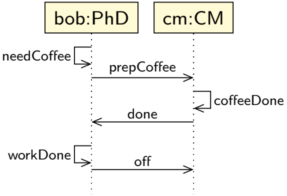
Sequence Diagram

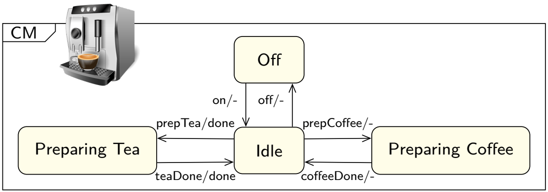
Lifeline conformance: For each lifeline, the sequence of received message symbols is a path of triggers in the attached state machine.

Formalization

Sequence Diagram

Lifeline conformance: For each lifeline, the sequence of received message symbols is a path of triggers in the attached state machine.

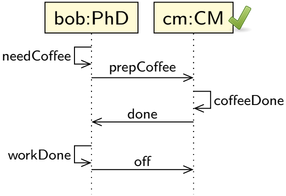


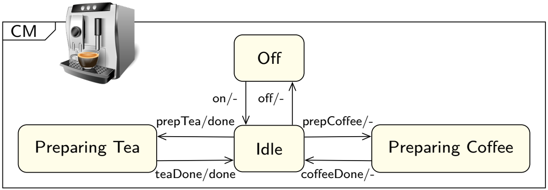


Formalization

Sequence Diagram

Lifeline conformance: For each lifeline, the sequence of received message symbols is a path of triggers in the attached state machine.

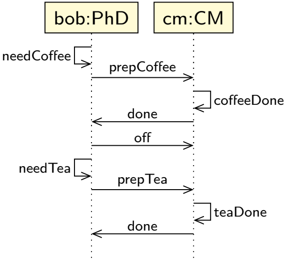


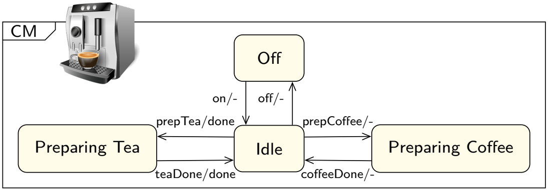


Formalization

Sequence Diagram

Lifeline conformance: For each lifeline, the sequence of received message symbols is a path of triggers in the attached state machine.

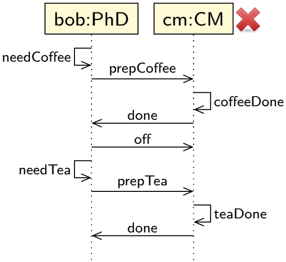


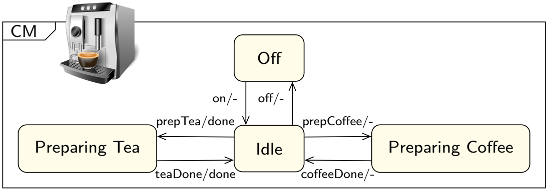


Formalization

Sequence Diagram

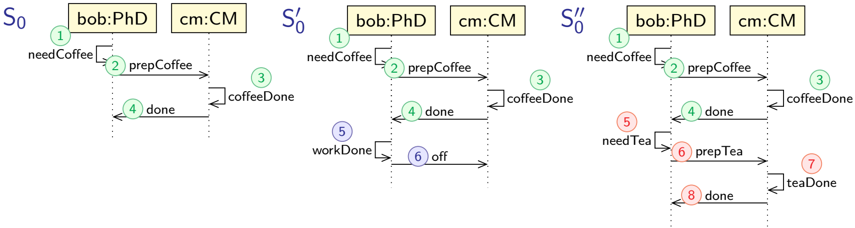
Lifeline conformance: For each lifeline, the sequence of received message symbols is a path of triggers in the attached state machine.





Problem statement

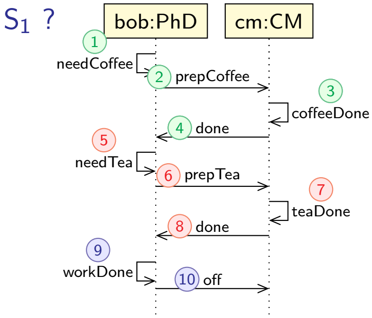
Instance: A sequence diagram and two of its *revisions*.

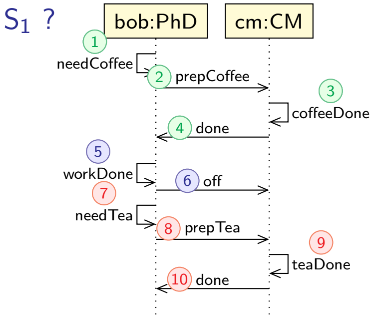


Problem statement

Objective: Find a *consolidated version*

- Contains all original and added messages and lifelines
- Lifelines conform to state machines





Guided Merging of Sequence Diagrams

Our Contributions

- Formalization of state machine, sequence diagram, formal problem statement
- **Translation to propositional SAT Problem**
- Implementation and evaluation

Translation to propositional SAT

Three types of variables

- m_i, m: message, i: position
- c_i^s, s: state in state machine, i: position, c: “source”
- t_i^s, s: state in state machine, i: position, t: “target”

Translation to propositional SAT

Three types of variables

- m_i, m: message, i: position
- c_i^s, s: state in state machine, i: position, c: "source"
- t_i^s, s: state in state machine, i: position, t: "target"

Propositional formula consists of constraints that describe legal solutions, e.g.

$$(m_1 \vee m_2 \vee m_3) \wedge (\neg m_1 \vee \neg m_2) \wedge (\neg m_2 \vee \neg m_3) \wedge (\neg m_1 \vee \neg m_3)$$

Translation to propositional SAT

Three types of variables

- m_i, m: message, i: position
- c_i^s, s: state in state machine, i: position, c: “source”
- t_i^s, s: state in state machine, i: position, t: “target”

Propositional formula consists of constraints that describe legal solutions, e.g.

$$(m_1 \vee m_2 \vee m_3) \wedge (\neg m_1 \vee \neg m_2) \wedge (\neg m_2 \vee \neg m_3) \wedge (\neg m_1 \vee \neg m_3)$$

$$m_1 \rightarrow ((c_1^{s1} \wedge t_1^{s2}) \vee (c_1^{s3} \wedge t_1^{s4}))$$

Translation to propositional SAT

Three types of variables

- m_i, m: message, i: position
- c_i^s, s: state in state machine, i: position, c: "source"
- t_i^s, s: state in state machine, i: position, t: "target"

Propositional formula consists of constraints that describe legal solutions, e.g.

$$(m_1 \vee m_2 \vee m_3) \wedge (\neg m_1 \vee \neg m_2) \wedge (\neg m_2 \vee \neg m_3) \wedge (\neg m_1 \vee \neg m_3)$$

$$m_1 \rightarrow ((c_1^{s1} \wedge t_1^{s2}) \vee (c_1^{s3} \wedge t_1^{s4}))$$

The encoding is polynomial in the input size.
Satisfying assignments of the formula can be directly translated back into a solution of our problem.

Translation to propositional SAT

$$\bigwedge_{m \in M} \left(\bigvee_{i \in \mathsf{allow}(m)} m_i \right) \wedge \bigwedge_{m \in M} \bigwedge_{\substack{i,j \in \mathsf{allow}(m) \\ i \neq j}} \left(\neg m_i \vee \neg m_j \right)$$

$$\bigwedge_{x \in \{o, \alpha, \beta\}} \bigwedge_{m \in M^x} \bigwedge_{i \in \mathsf{allow}(m)} \left(\neg m_i \vee \bigvee_{\substack{n \in M^x, \\ n \succ m}} \bigvee_{\substack{j > i, \\ j \in \mathsf{allow}(n)}} n_j \right)$$

$$\bigwedge_{i \in \mathsf{allow}(m)} \left(\neg m_i \vee \bigvee_{t \in \mathsf{trans}(m)} \left(c_i^{\pi_1(t)} \wedge t_i^{\pi_4(t)} \right) \right)$$

$$\bigwedge_{i=1}^{k} \left(\left(\bigvee_{c_i^s \in \mathsf{vc}} c_i^s \right) \wedge \left(\bigvee_{t_i^s \in \mathsf{vt}} t_i^s \right) \wedge \bigwedge_{s \in S_{all}} \bigwedge_{r \in S_{all} \setminus s} \left((\neg c_i^s \vee \neg c_i^r) \wedge (\neg t_i^s \vee \neg t_i^r) \right) \right)$$

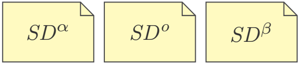
$$\bigwedge_{i=1}^{k-1} \bigwedge_{M \in \mathcal{SM}} \bigwedge_{s \in \pi_1(SM)} \left(\left(t_i^s \rightarrow \bigwedge_{r \in \pi_1(SM) \setminus s} \neg c_{i+1}^r \right) \wedge \left(\bigwedge_{j=1}^{i} \left(t_i^s \wedge \bigwedge_{l=1}^{j} \neg c_l^s \rightarrow \bigwedge_{r \in \pi_1(SM) \setminus s} \neg c_{j+1}^r \right) \right) \right)$$

Guided Merging of Sequence Diagrams

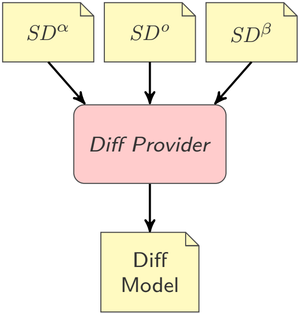
Our Contributions

- Formalization of state machine, sequence diagram, formal problem statement
- Translation to propositional SAT Problem
- **Implementation and evaluation**

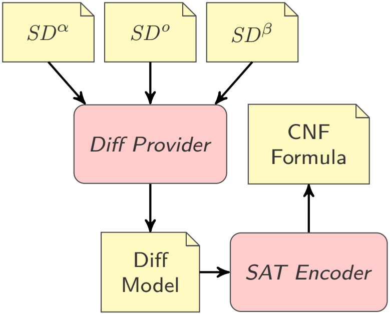
Implementation – Workflow



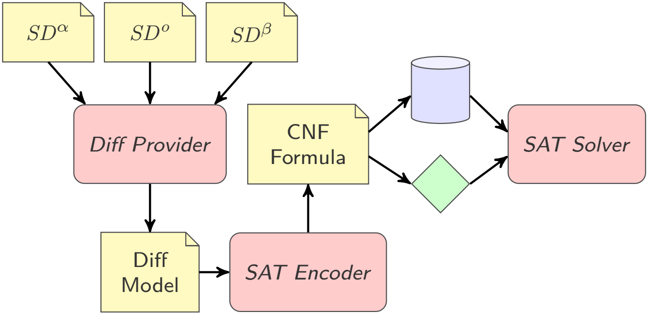
Implementation – Workflow



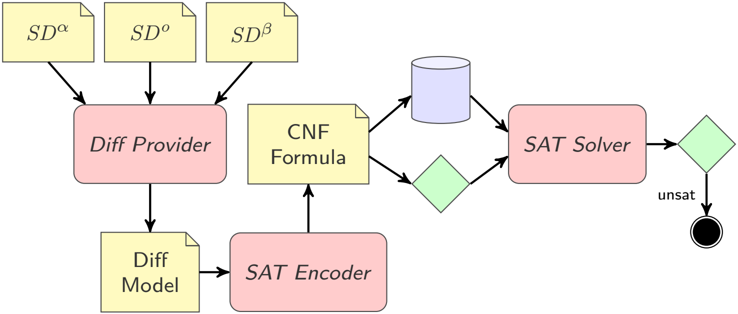
Implementation – Workflow



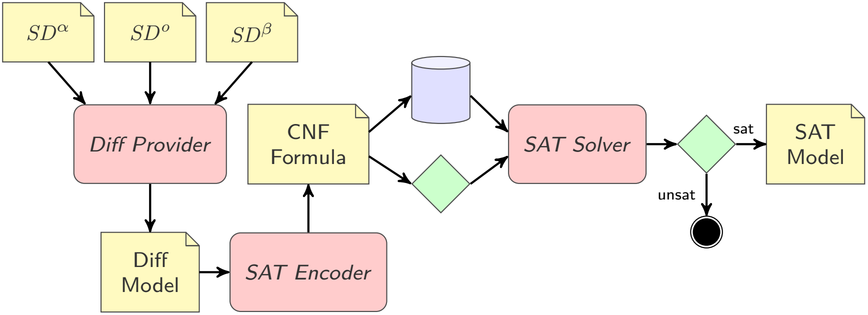
Implementation – Workflow



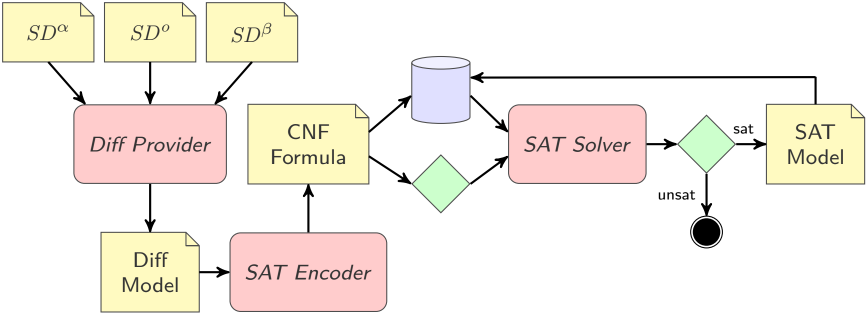
Implementation – Workflow



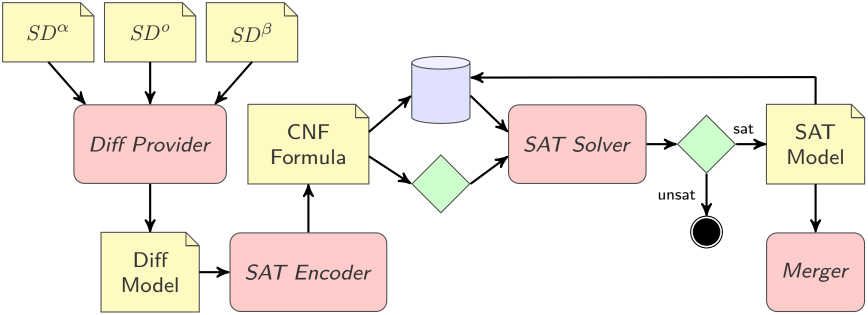
Implementation – Workflow



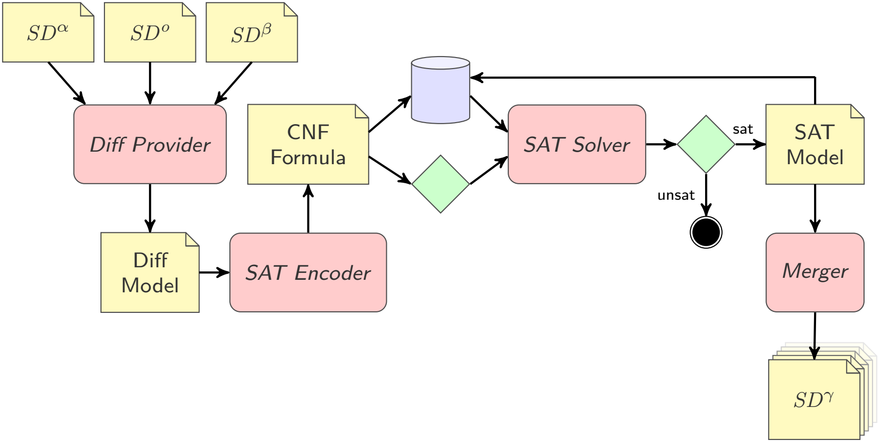
Implementation – Workflow



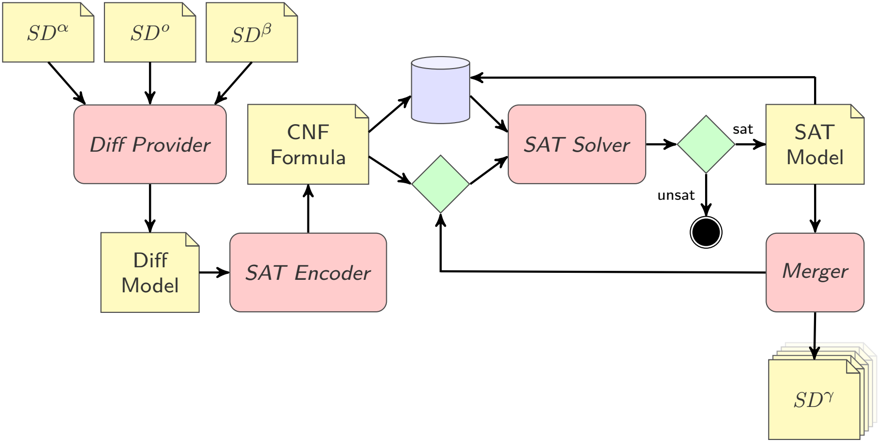
Implementation – Workflow



Implementation – Workflow



Implementation – Workflow



Evaluation

- Benchmark set with 45 instances

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined
- Quit after 1,000 solutions are found

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined
- Quit after 1,000 solutions are found

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined
- Quit after 1,000 solutions are found

Set	# SM	# action symbols	# states	# transitions
email	3	15	16	19
coffee	2	9	7	8
philosopher	2	8	7	8

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined
- Quit after 1,000 solutions are found

Set	# SM	# action symbols	# states	# transitions
email	3	15	16	19
coffee	2	9	7	8
philosopher	2	8	7	8

Results

- Between 0.06s and 0.2s per solution depending on instance

Evaluation

- Benchmark set with 45 instances
- For some instances, no state machines are defined
- Quit after 1,000 solutions are found

Set	# SM	# action symbols	# states	# transitions
email	3	15	16	19
coffee	2	9	7	8
philosopher	2	8	7	8

Results

- Between 0.06s and 0.2s per solution depending on instance
- Some instances have many solutions (>1,000)

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

What we consider next

- Handling high numbers of solutions

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

What we consider next

- Handling high numbers of solutions
- Including deletions and updates

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

What we consider next

- Handling high numbers of solutions
- Including deletions and updates
- Integration of other UML concepts

Summary and Future Work

We

- Formalized a subset of the UML in our language *tMVML*
- Translated the sequence diagram merging problem to prop. SAT
- Implemented and evaluated our approach

What we consider next

- Handling high numbers of solutions
- Including deletions and updates
- Integration of other UML concepts
- Visualization

Related Work

Model merging:

- *Gerth et al.*, Merge support for business process models using term rewriting systems
- *Cicchetti et al.*, Definition of conflict patterns
- *Nejati et al.*, Merging of state machines

Consistency checking:

- *Diskin et al.*, Category theory based framework
- *Van der Straeten et al.*, Inconsistency detection between class and sequence diagrams using Kodkod
- *Sabatzadeh et al.*, Consistency checks between overlapping models
- *Tsoliakis*, Integration of constraints of other views into sequence diagrams
- *Brosch et al.*, Model checking on state machines and sequence diagrams