A Manifesto for Model Merging

Authors:Greg Brunet, Marsha Chechik, Steve Easterbrook, Shiva Nejati, Nan Niu, Mehrdad Sabetzadeh

Presented by : Preksha Sisodia

Contents

Contents

Problem Description

- How do we compare and evaluate merging techniques?
- Framework for Comparison of Merging Techniques
- Need for Framework
  - Existing approaches make assumption about
    - Types of model being merged
    - Relationship between them

Contents

Background

- What is Model Merging?
- Model Merging Techniques
  - ERD merging(structural model merge)
  - State machine model merging(behavioral model merge)

Contents

Challenges in Model Merging

- Use of different vocabularies
- Models may overlap
- Source models are updated
- Heterogeneous

How Does The Framework Help?

- Need a precise way to state the relationship between models and make them first-class artifacts

- To understand the properties of model and model relationships

Contents

The Framework

1. Data Types used in the definition
2. Model Management Operators
3. Algebraic Properties

- Assumptions

  models to be merged are similar

Contents

The Framework – Data Types

- Models
- Relationships
- Properties of models

Contents

The Framework - Model Management Operators(MMO)

merge : *model* × *model* × *relationship* → *model*

- Merges models

match: *model* × *model* → *relationship*

- Gives the commonalities between the models

diff: *model* × *model* → *transformation*

Diff : *m1* × *m2* → *t1*

Diff : *m2* × *m1*→ *t2*

Contents

Model Management Operators(Contd..)

split: $model \rightarrow model \times model \times relationship$

- Produces compatible models

slice: $model \times criterion \rightarrow model$

- Partial view of a model based on criterion

check-property : $model \times property \rightarrow truth\text{-}value$

Contents

Other Model Management Operators(Contd..)

is-consistent: *model* × *model* × *relationship* → *truth-value*

patch: *model* × *transformation* → *model*

propagate : *transformation* × *model* × *model* × *relationship* → *model*

Contents

The Framework - Algebraic Properties

- Idempotency

  merge(m1,m1,match(m1,m1)) = m1
- Commutative

  merge(m1,m2, r) = merge(m2,m1, r)
- Associative

  merge(merge(m1,m2, r1,2),m3, r(1,2),3) = merge(m1, merge(m2,m3, r2,3), r1,(2,3))
- Inverse

  split(merge(m1,m2, r)) = (m1,m2, r)

Contents

The Framework - Algebraic Properties(Contd..)

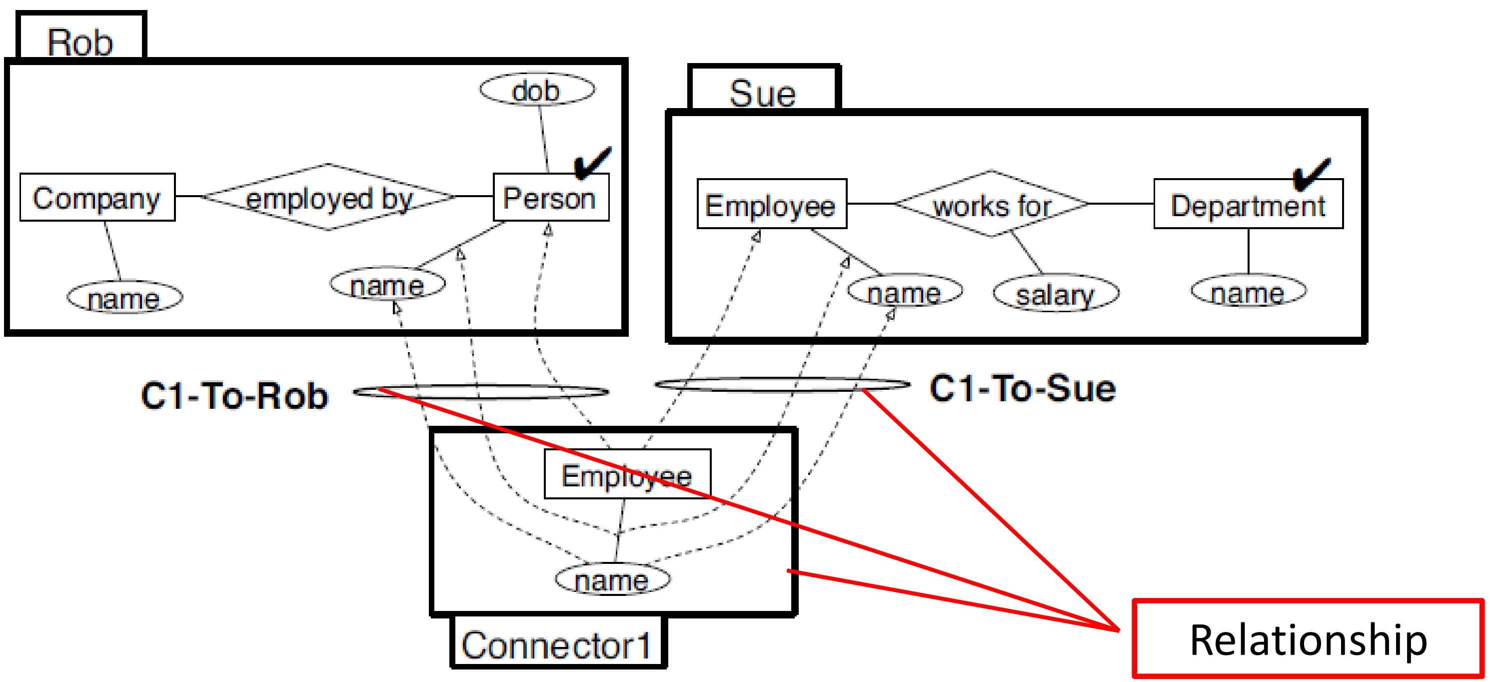
- Monotonicity:
  - *merge (m1, m2, relationship) => m*
  - *merge (m1',m2', relationship) = > m'*
  - *Where m1' evolves from m1 and m2' evolves from m2*
  - *Then m' evolves from m*
- Totality : $\forall m1, m2 \in model \cdot merge(m1, m2, r) \in model$

Applying The Framework

Contents

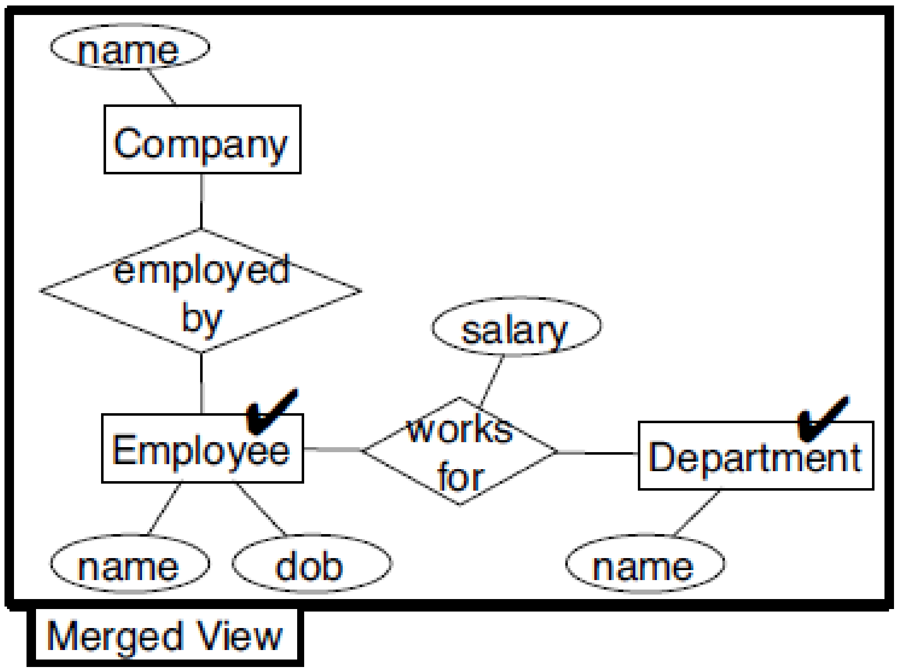
- Two Examples:
  - Structural Model Merging (ERDs)
  - Behavioral Model Merging (State Machines)

Merging structural model for Payroll Application



(a) Interrelating the perspectives

Merged structural model



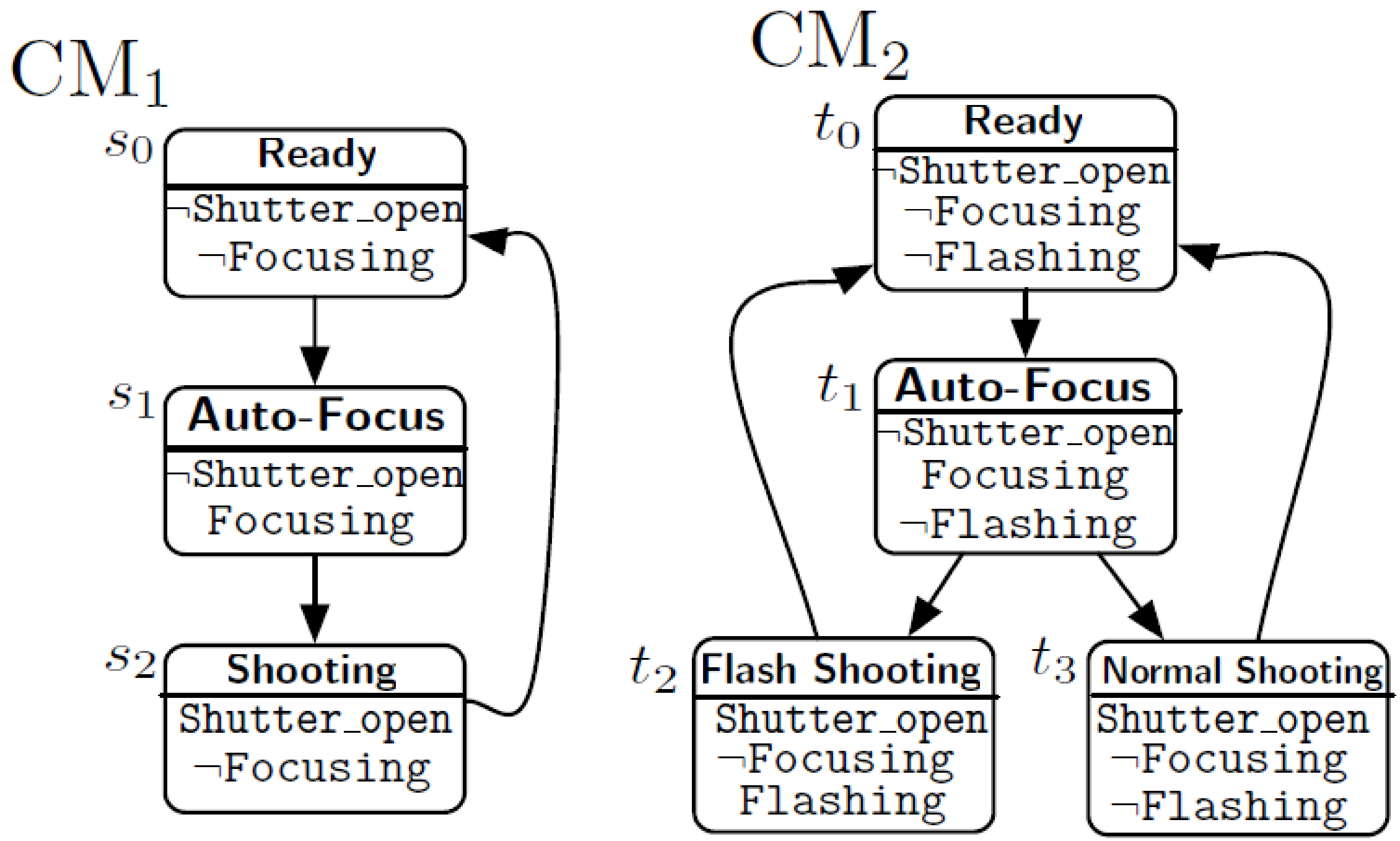
(b) The merged model

Contents

Applying the Framework to Structural Merging

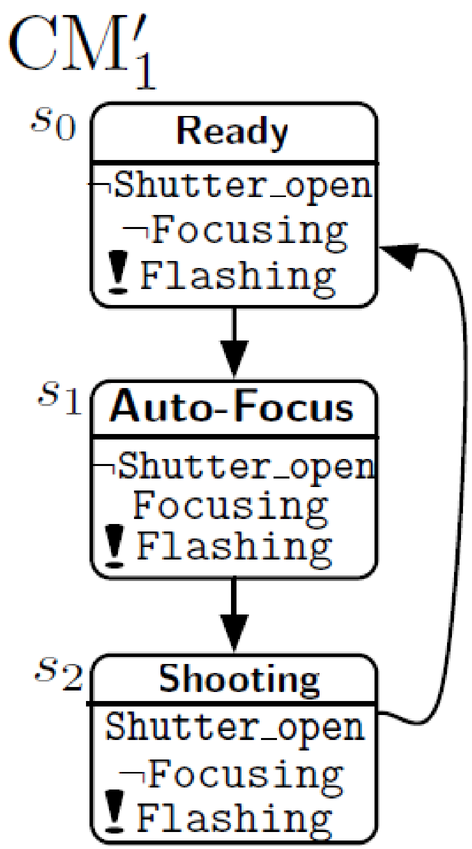
- Relationship - connector1, C1-To-Rob, C1-To-Sue
- Follows Idempotency, Commutative, Associative
- Inverse - if traceability information is stored
- Monotonicity - if evolution is considered as model inclusion
- Totality – embedding preserves syntactic constraints

Merging Behavioral Model for Camera Application

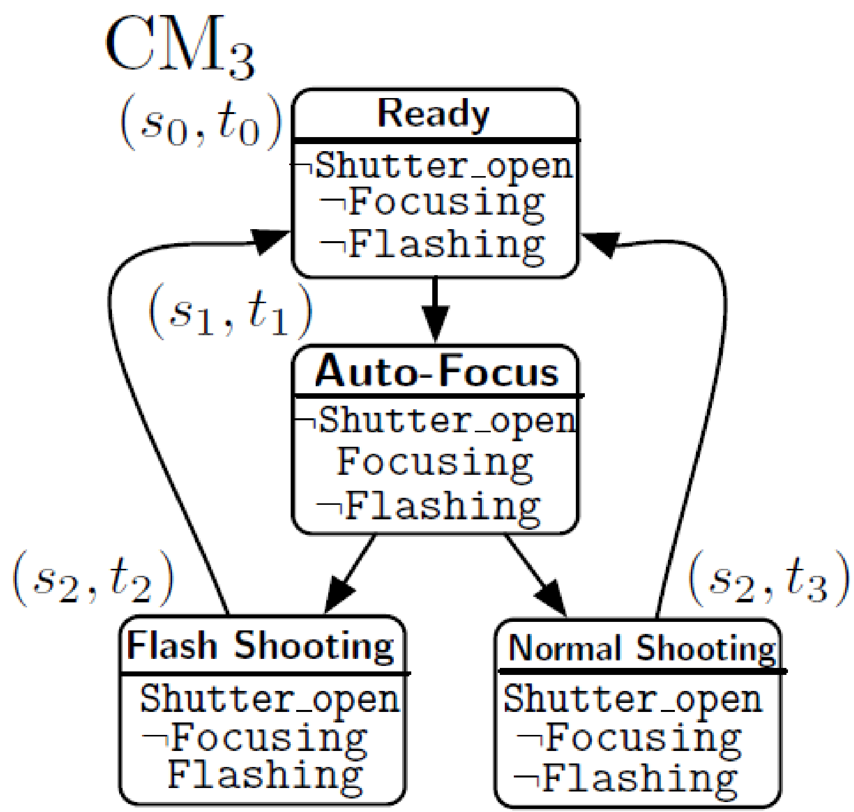


Merging Behavioral Model For Payroll Application(Contd..)

Unifying variables



Merged Model



Contents

Applying The Framework To Behavioral Merging

- Relationship between CM1 and CM2 is *{(s0, t0), (s1, t1), (s2, t2), (s2, t3)}*
- Follows Idempotency, Commutative, Associative
- Inverse - if traceability information is stored
- Monotonicity - if evolution is considered as refinement relation over models
- Totality – Does not hold. Only consistent models can be merged

Comparison of Two Merging Techniques

	Structural Model Merging	**Behavioral Model Merging**
Idempotency	✓	✓
Commutativity	✓	✓
Associativity	✓	✓
Inverse[1]	✓	✓
Monotonicity[2]	✓[2]	✓[3]
Totality	✓	✗

1 - With proper traceability info
2 - if model inclusive
3 - if there is refinement relation over models

Contents

Conclusion

- Framework to provide a way to compare model merging techniques
- Play around with models using operators like split, slice, propagate, etc.
- By treating relationships as a data type, they are give equal importance as models
- Implicit determination of relationships sometime is not what the user wants, in those cases this framework helps in stating the relationships explicitly the way the user prefers.

References

- M. Sabetzadeh and S. Easterbrook. “An Algebraic Framework for Merging Incomplete and Inconsistent Views”.
- Mehrdad Sabetzadeh and Shiva Nejati.“TReMer: A Tool for Relationship-Driven Model Merging”

Discussion points

- Only homogenous models were discussed.
- Why are these algebraic properties important in a merging technique
- Is the framework scalable to any level?
- What about domain specific algebraic properties
- Other operators like split, slice, propagate etc.
- When we are trying to match models what features are we exactly exploring