UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

HØGSKOLEN
I BERGEN
BERGEN UNIVERSITY COLLEGE

Integration of Runtime Verification into Metamodeling for Simulation and Code Generation (Position Paper)

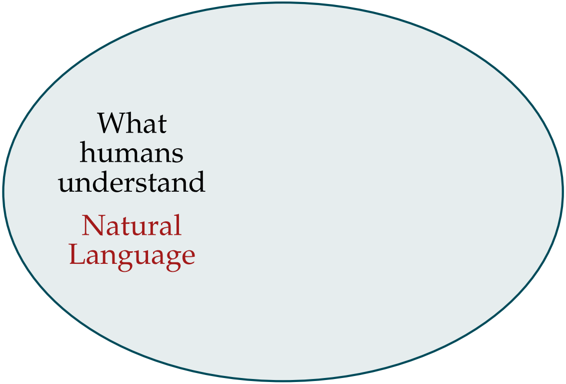
F. Macias T. Scheffel M. Schmitz R. Wang

{fernando.macias,rui.wang}@hib.no

{scheffel,schmitz}@isp.uni-luebeck.de

16th International Conference on Runtime Verification, September 2016, Madrid, Spain

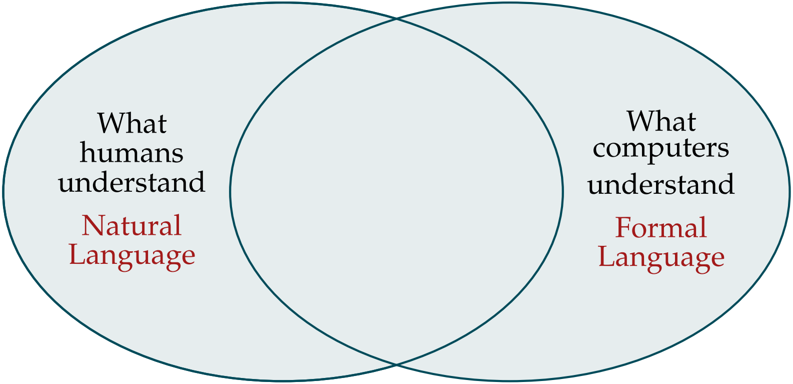
Why Modeling?



What humans understand

Natural Language

Why Modeling?



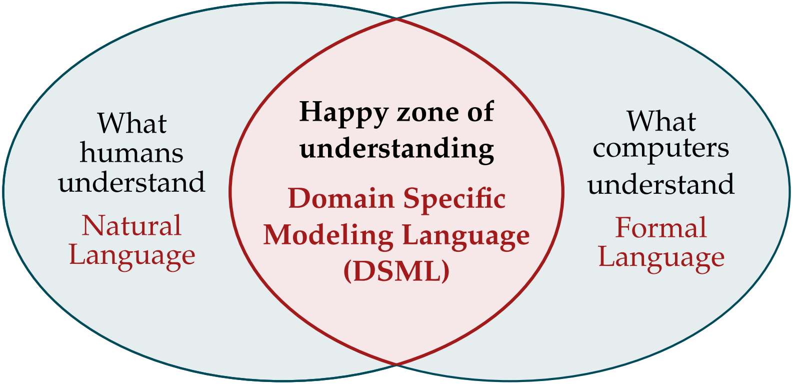
What humans understand

Natural Language

What computers understand

Formal Language

Why Modeling?



What humans understand

Natural Language

Happy zone of understanding

Domain Specific Modeling Language (DSML)

What computers understand

Formal Language

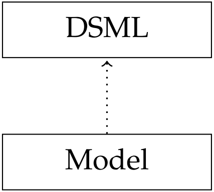
Runtime Verification

- DSMLs do not shield the software from design errors
- Testing is seldom exhaustive
- Model Checking can be performed on most models, but how to detect the unexpected on target platforms?
- Runtime Verification checks execution of real system
  - Consider environmental influences
  - React to failures

Integrating RV into Modeling

Goal

⇒ Integrate verification into software engineering process.
⇒ Domain experts and verification engineers use same model.



DSML

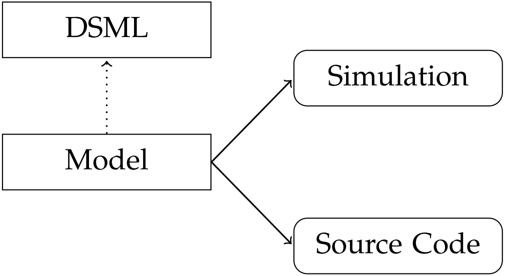
Model

Integrating RV into Modeling

Goal

⇒ Integrate verification into software engineering process.

⇒ Domain experts and verification engineers use same model.

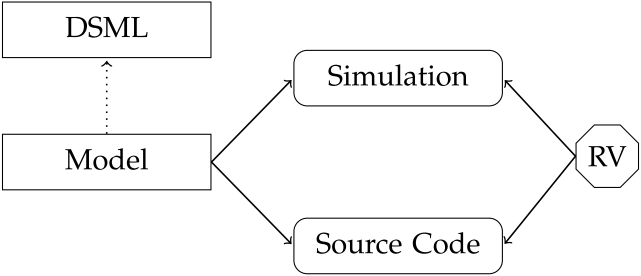


Integrating RV into Modeling

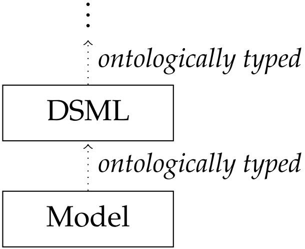
Goal

⇒ Integrate verification into software engineering process.

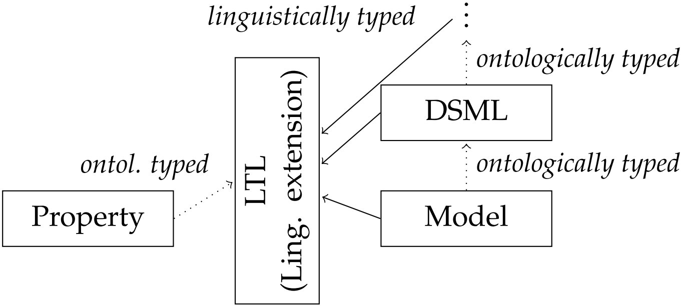
⇒ Domain experts and verification engineers use same model.



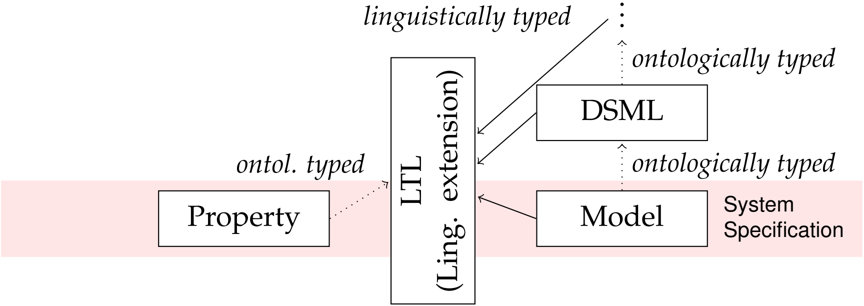
Multilevel Metamodeling



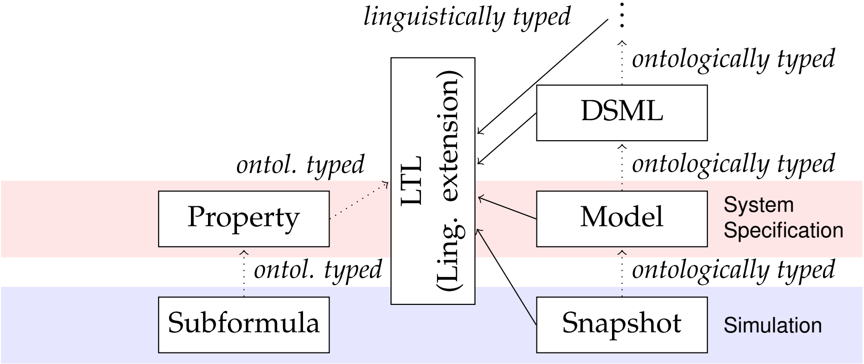
Multilevel Metamodeling



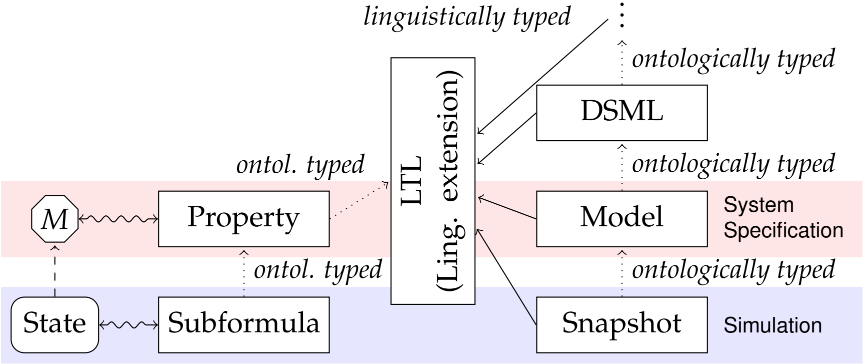
Multilevel Metamodeling



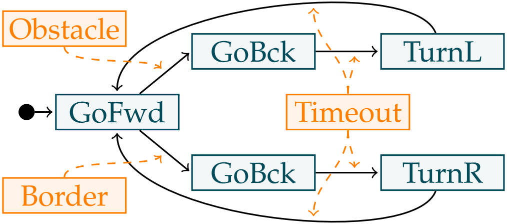
Multilevel Metamodeling



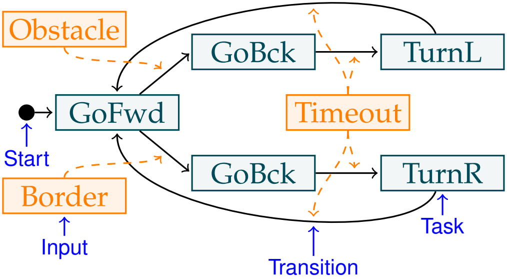
Multilevel Metamodeling



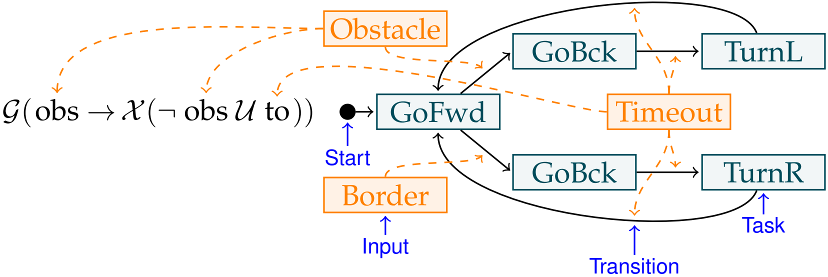
Example Scenario



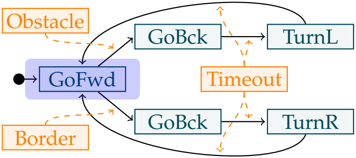
Example Scenario



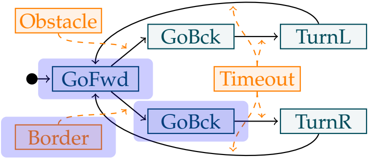
Example Scenario

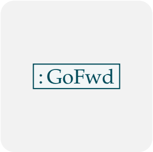

$\mathcal{G}(\mathrm{obs} \rightarrow \mathcal{X}(\neg\,\mathrm{obs}\,\mathcal{U}\,\mathrm{to}))$
Obstacle
Timeout
Border
GoFwd
GoBck
TurnL
GoBck
TurnR
Start
Input
Transition
Task

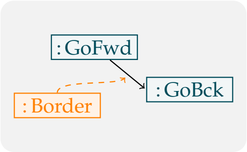
Example Simulation



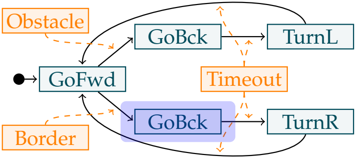
Example Simulation

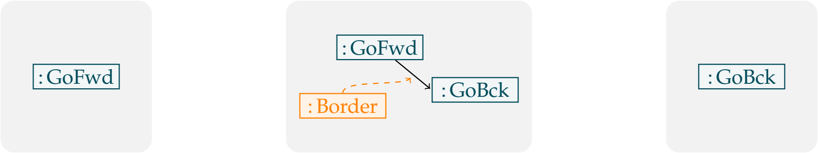




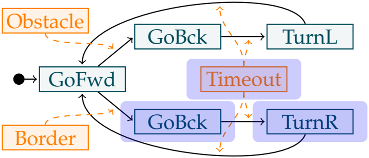


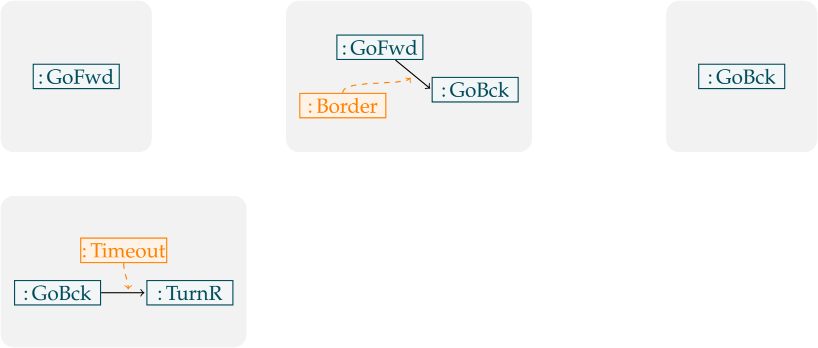
Example Simulation



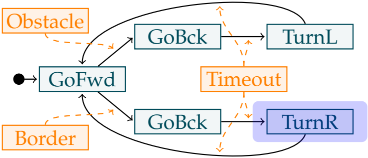


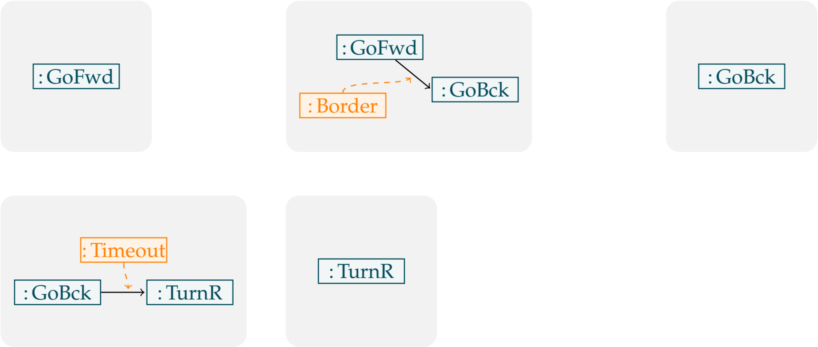
Example Simulation



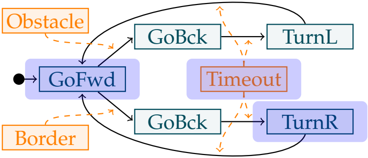


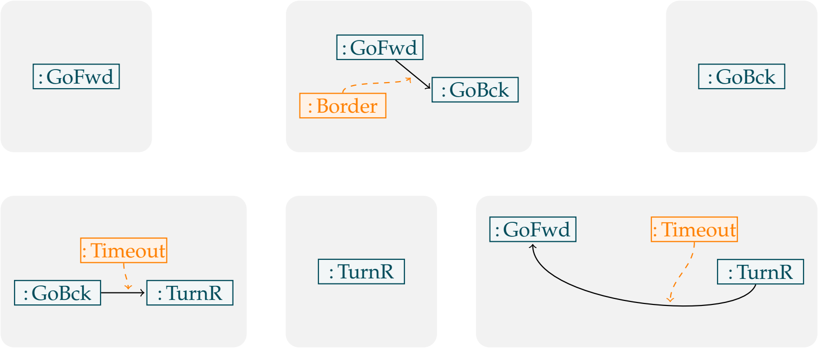
Example Simulation



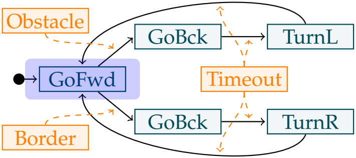


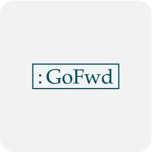
Example Simulation





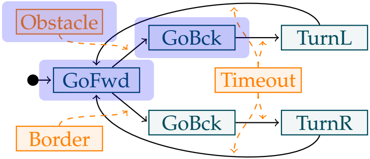
Example Simulation II

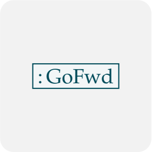




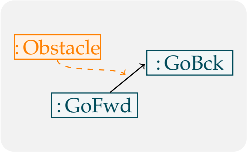
$\mathcal{G}(\mathrm{obs} \rightarrow \mathcal{X}(\neg\,\mathrm{obs}\,\mathcal{U}\,\mathrm{to}))$

Example Simulation II



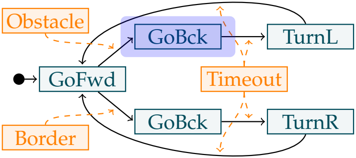


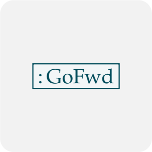
$\mathcal{G}(\text{obs} \rightarrow \mathcal{X}(\neg\, \text{obs}\, \mathcal{U}\, \text{to}))$



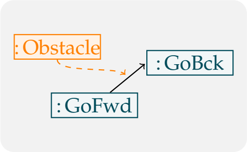
$\mathcal{G}(\text{obs} \rightarrow \mathcal{X}(\neg\, \text{obs}\, \mathcal{U}\, \text{to}))$

Example Simulation II

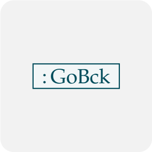




$\mathcal{G}(\text{obs} \rightarrow \mathcal{X}(\neg\,\text{obs}\,\mathcal{U}\,\text{to}))$

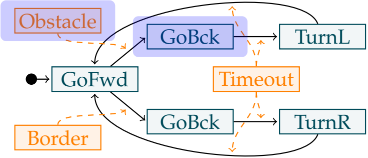


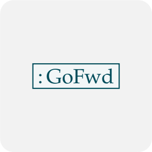
$\mathcal{G}(\text{obs} \rightarrow \mathcal{X}(\neg\,\text{obs}\,\mathcal{U}\,\text{to}))$



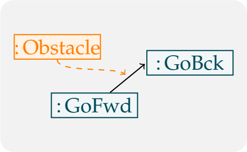
$\neg\,\text{obs}\,\mathcal{U}\,\text{to}$

Example Simulation II

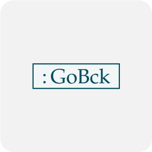




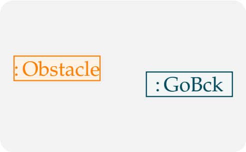
$\mathcal{G}(\text{obs} \to \mathcal{X}(\neg\,\text{obs}\,\mathcal{U}\,\text{to}))$



$\mathcal{G}(\text{obs} \to \mathcal{X}(\neg\,\text{obs}\,\mathcal{U}\,\text{to}))$

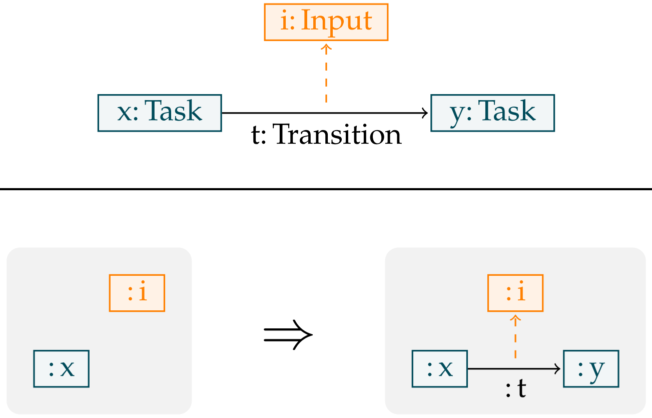


$\neg\,\text{obs}\,\mathcal{U}\,\text{to}$

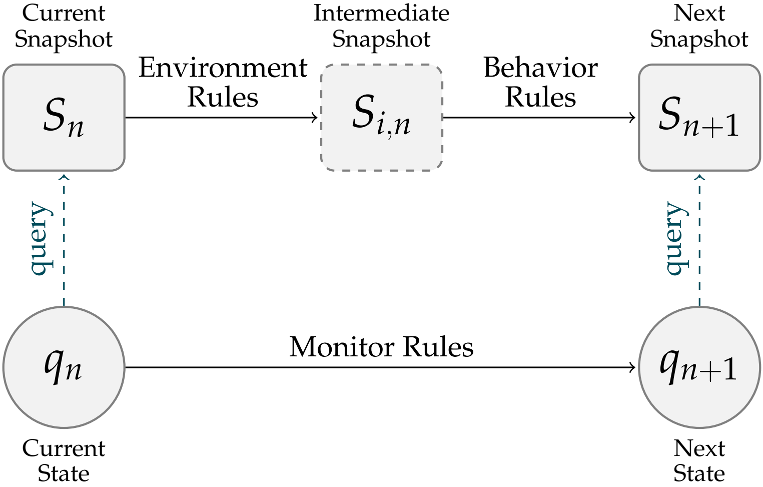


$\neg\,\text{obs}\,\mathcal{U}\,\text{to}$

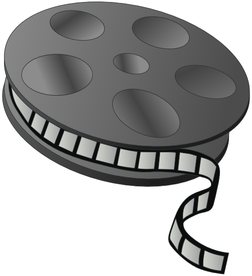
Coupled Model Transformation Rules



Model Transformation Rules



Code Generation



Conclusion

- Synchronize correctness property specification with behavioral model
- Check correctness of behavior model in
  - Simulation through coupled model transformation and
  - Target hardware through code generation.
- Integrate verification into early stages of development and check same properties on deployed software.

Outlook

- More Formalization
  - Coupled Model Transformation Rules
- More Application
  - Distributed Systems
  - Healthcare workflows
- More Expressiveness
  - Time
  - Data