Formally Verified Animation for RoboChart Using Interaction Trees

Kangfeng Ye, Simon Foster, Jim Woodcock

robostar.cs.york.ac.uk

October 26, 2022

Outline

Background and motivations

RoboChart model

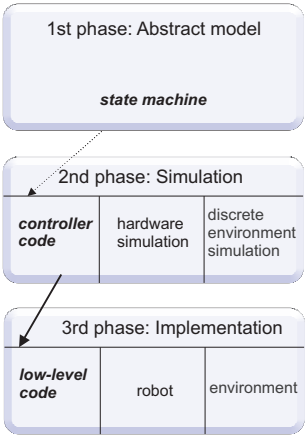
Animation of RoboChart

Operational Semantics for RoboChart

Conclusion

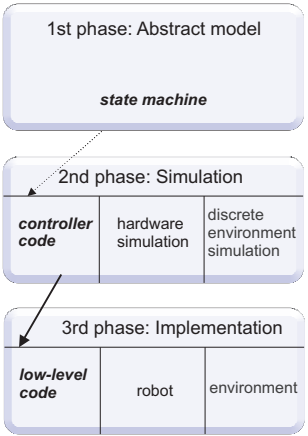
ROBOSTAR

Current practice and problems



ROBOSTAR

Current practice and problems



- No models, or models without precise syntax or formal semantics,
- Time and uncertainty: discussed informally,
- No tool support,
- Loose connections of artefacts,
- Trial-and-error,
- No assurance.

ROBOSTAR

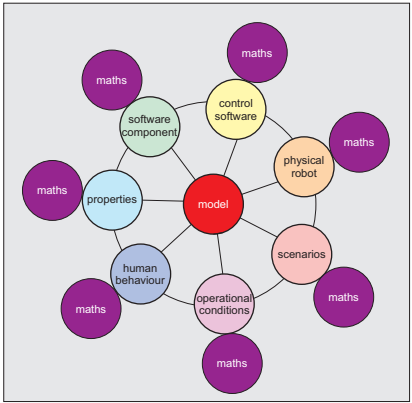
RoboStar

- RoboStar framework: modern modelling and verification technologies, software engineering of robotics

ROBOSTAR

RoboStar

- RoboStar framework: modern modelling and verification technologies, software engineering of robotics
- Vision: model centred, mathematical semantics



ROBOSTAR

RoboChart

- Core notation of RoboStar, DSL for robotics, state machines + time + probability

ROBOSTAR

RoboChart

- Core notation of RoboStar, DSL for robotics, state machines + time + probability
- A component model (platform independent + parallel composition of state machines)

ROBOSTAR

RoboChart

- Core notation of RoboStar, DSL for robotics, state machines + time + probability
- A component model (platform independent + parallel composition of state machines)
- Denotational semantics (CSP and MDP) and verification (FDR and PRISM)

ROBOSTAR

RoboChart

- Core notation of RoboStar, DSL for robotics, state machines + time + probability
- A component model (platform independent + parallel composition of state machines)
- Denotational semantics (CSP and MDP) and verification (FDR and PRISM)
- Animated available in FDR (ProBE animator), or in PRISM (simulator)

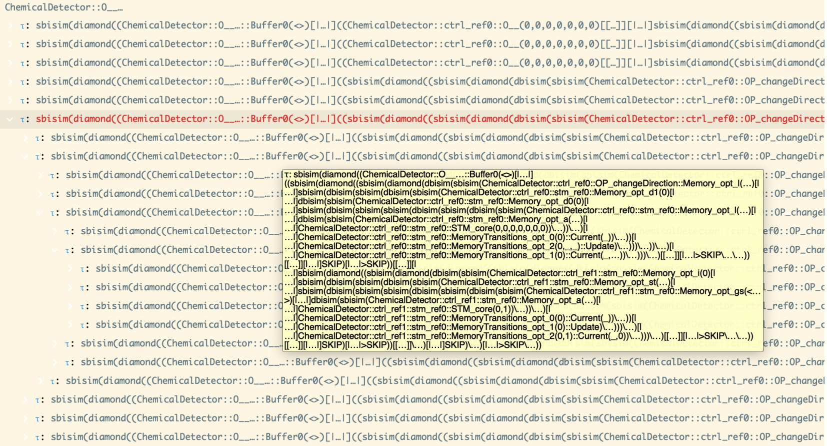
ROBOSTAR

RoboChart

- Core notation of RoboStar, DSL for robotics, state machines + time + probability
- A component model (platform independent + parallel composition of state machines)
- Denotational semantics (CSP and MDP) and verification (FDR and PRISM)
- Animated available in FDR (ProBE animator), or in PRISM (simulator)
- but the animation is low-level and not very helpful (for our purpose)

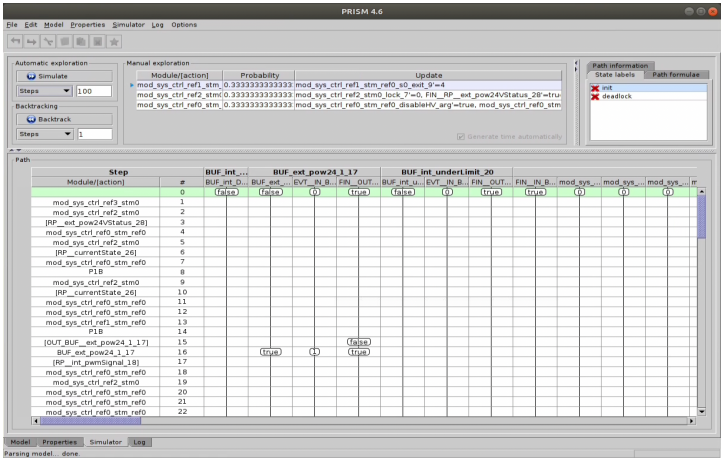
ROBOSTAR

Animation using FDR



ROBOSTAR

Animation using PRISM



ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),
- Suitable for early iterations of developing models,

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),
- Suitable for early iterations of developing models,
- Demonstration: insight into models and implicit assumptions,

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),
- Suitable for early iterations of developing models,
- Demonstration: insight into models and implicit assumptions,
- Understanding of behaviour of robot controllers in particular scenarios,

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),
- Suitable for early iterations of developing models,
- Demonstration: insight into models and implicit assumptions,
- Understanding of behaviour of robot controllers in particular scenarios,
- Interactive testing of models and properties,

ROBOSTAR

Why Animation?

(Kazmierczak et al. 1998)

- Easily automated and cheap to perform (vs. formal verification),
- Require little expertise (vs. model-checking and theorem proving),
- Suitable for early iterations of developing models,
- Demonstration: insight into models and implicit assumptions,
- Understanding of behaviour of robot controllers in particular scenarios,
- Interactive testing of models and properties,
- Shorten learning curve of RoboChart (semantics): students, roboticists, formal experts.

ROBOSTAR

Recent work

Foster, S., Hur, C.K., Woodcock, J.: Formally verified simulations of state-rich processes using interaction trees in Isabelle/HOL. CONCUR (2021)

- Interaction trees (ITrees) in Isabelle/HOL, with ITrees-based semantics for CSP,
- Formally verified animation for CSP (code generator in Isabelle/HOL),
- Implemented most CSP processes and operators.

ROBOSTAR

Contributions

- Operational semantics for RoboChart: ITrees-based CSP,
- Mechanisation of the semantics of the autonomous chemical detector model in Isabelle,
- Animation of the model,
- Implementation of three extra CSP operators (interrupt, exception, and renaming),
- Implementation of a bounded sequence type for code generation.

ROBOSTAR

Contributions

- Operational semantics for RoboChart: ITrees-based CSP,
- Mechanisation of the semantics of the autonomous chemical detector model in Isabelle,
- Animation of the model,
- Implementation of three extra CSP operators (interrupt, exception, and renaming),
- Implementation of a bounded sequence type for code generation.

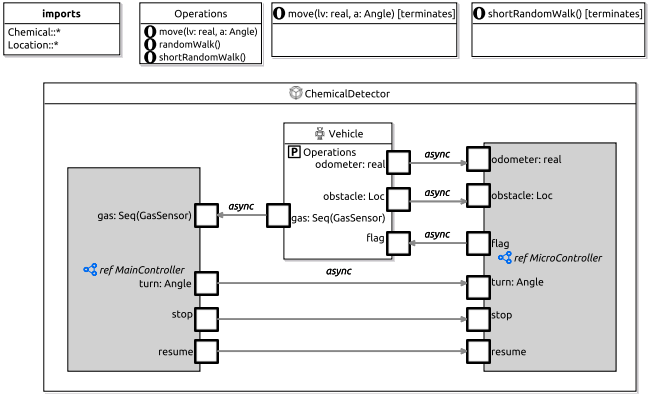
Benefits

- Unification of verification and animation,
- Support richer RoboChart types and expressions, and functions,
- Characterise systems with an infinite number of states symbolically,
- Functional algorithms and data refinement: automated,
- Could be fully automated: from RoboChart models to final Haskell code.

ROBOSTAR

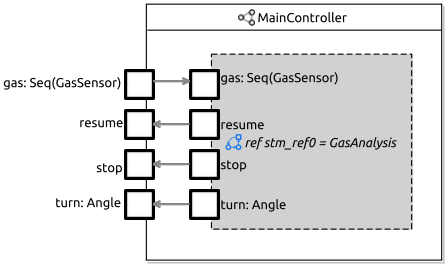
Autonomous chemical detector - module

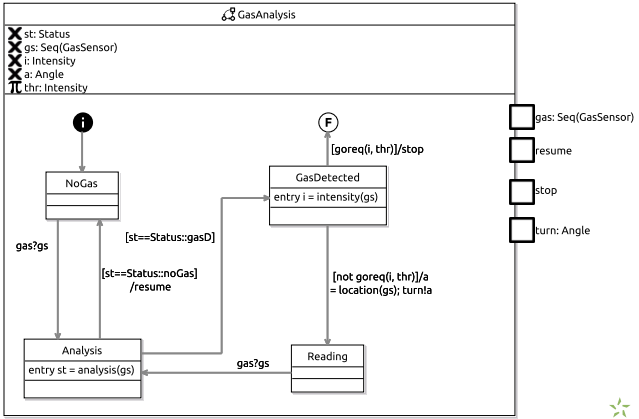
From the papers (Hilder et al. 2012) and (Miyazawa et al., 2019).



ROBOSTAR

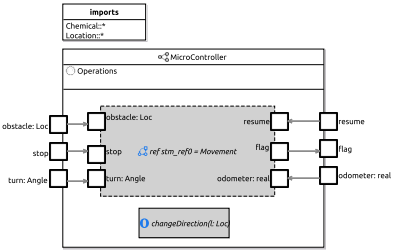
Autonomous chemical detector - main controller

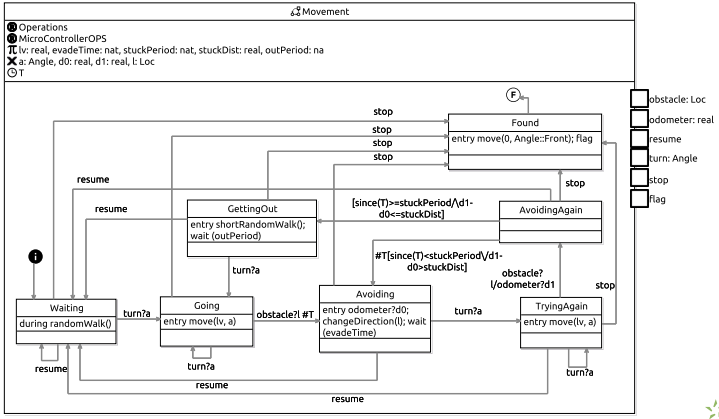




ROBOSTAR

Autonomous chemical detector - micro controller



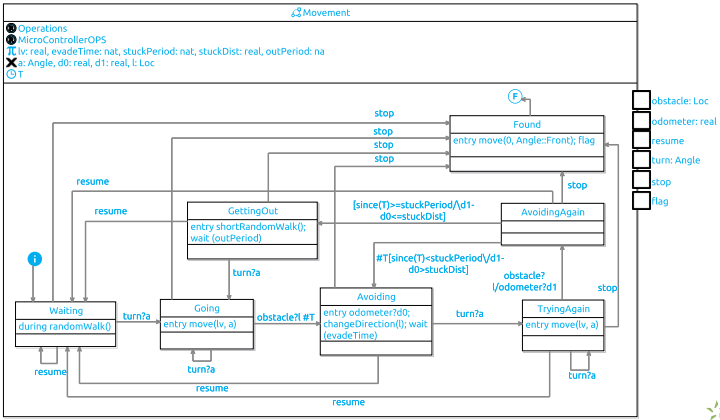


ROBOSTAR

Animation of RoboChart (video)

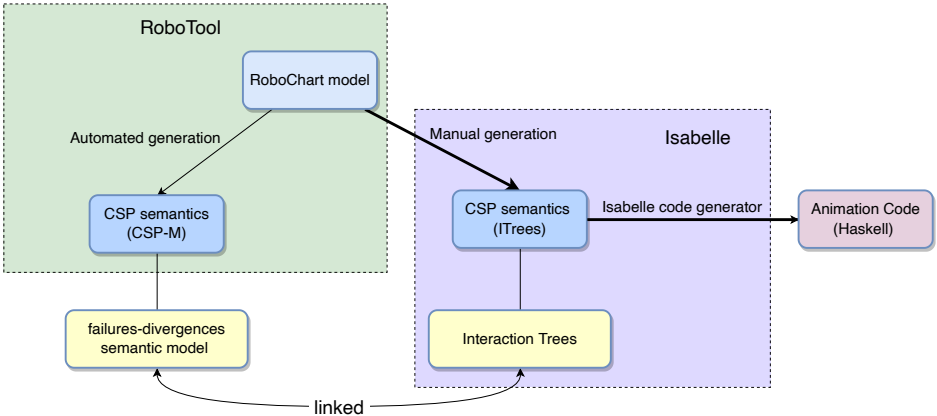
- Model
- Controllers
- State machines

Code generation in Isabelle, animation (Isabelle console or OS console)



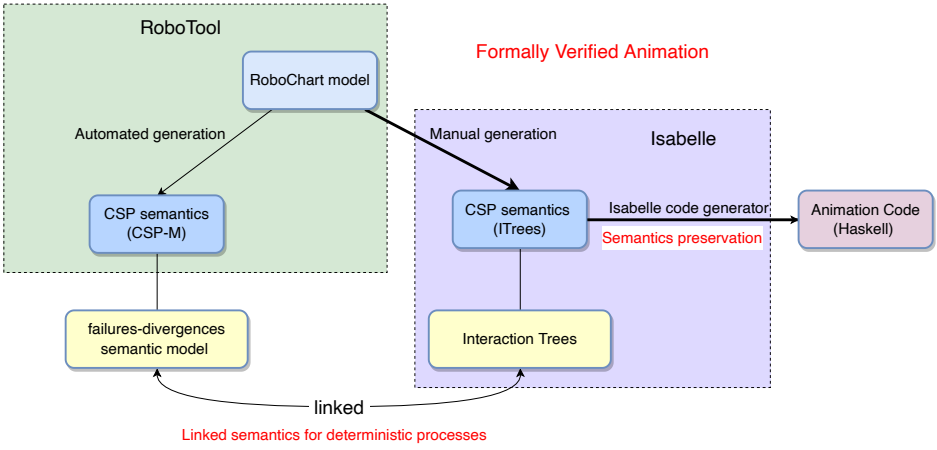
ROBOSTAR

Animation of RoboChart - our approach



ROBOSTAR

Animation of RoboChart - our approach (semantics)



Interaction Trees (Xia et al. 2019) in Isabelle/HOL

Coinductive trees, with potentially infinite breadth and depth, used to represent the ways a process communicates with its environment and evolves over time.

Definition (Interaction trees)

```
codatatype ('e, 'r) itree =
  Ret 'r | - ‹ Terminate, returning a value ›
  Sil "('e, 'r) itree" | - ‹ Invisible event ›
  Vis "'e ↛ ('e, 'r) itree" - ‹ Visible events and continuations ›
```

Notation: $\checkmark_v$ for `Ret` v, τP for `Sil` P, $[\!]\ e \in E \to P(e)$ for `Vis` $(\lambda\, e \in E \bullet P(e))$.

ROBOSTAR

Existing CSP Processes and Operators

- *skip*, *stop*, *div*, *run*
- Input event: *inp_in* $c\ A$
- Output event: *outp* $c\ v$
- Sequential composition (monad): $P \gg\!= K$
  - *do* $\{x \leftarrow$ *inp_in* $c\ A;$ *outp* $d\ (2 \cdot x);$ *Ret* $x\}$
- External choice: $P \Box Q$
- Parallel composition: $P \parallel_A Q$ (interleave $|||$).
- Hiding: $P \setminus A$

ROBOSTAR

Extra CSP operators - Interrupt

Definition (Interrupt)

$P \triangle Q$ behaves like P except that if at any time Q performs one of its initial events then it takes over, defined corecursively.

ROBOSTAR

Extra CSP operators - Interrupt

Definition (Interrupt)

$P \triangle Q$ behaves like P except that if at any time Q performs one of its initial events then it takes over, defined corecursively.

$$(\mathit{Sil}\, P') \triangle Q = \mathit{Sil}\, (P' \triangle Q) \qquad P \triangle (\mathit{Sil}\, Q') = \mathit{Sil}\, (P \triangle Q')$$

$$(\mathit{Ret}\, x) \triangle Q = \mathit{Ret}\, x \qquad P \triangle (\mathit{Ret}\, x) = \mathit{Ret}\, x$$

$$(\mathit{Vis}\, F) \triangle (\mathit{Vis}\, G) = \mathit{Vis}\, \big(\{e \mapsto (P' \triangle Q) \mid (e \mapsto P') \in (\mathsf{dom}(G) \lhd F)\} \oplus G \big)$$

ROBOSTAR

Extra CSP operators - Interrupt

Definition (Interrupt)

$P \triangle Q$ behaves like P except that if at any time Q performs one of its initial events then it takes over, defined corecursively.

$$(\mathit{Sil}\, P') \triangle Q = \mathit{Sil}\, (P' \triangle Q) \qquad P \triangle (\mathit{Sil}\, Q') = \mathit{Sil}\, (P \triangle Q')$$

$$(\mathit{Ret}\, x) \triangle Q = \mathit{Ret}\, x \qquad P \triangle (\mathit{Ret}\, x) = \mathit{Ret}\, x$$

$$(\mathit{Vis}\, F) \triangle (\mathit{Vis}\, G) = \mathit{Vis}\, \big(\{e \mapsto (P' \triangle Q) \mid (e \mapsto P') \in (\mathsf{dom}(G) \lhd F)\} \oplus G \big)$$

ROBOSTAR

Extra CSP operators - Interrupt

Definition (Interrupt)

$P \triangle Q$ behaves like P except that if at any time Q performs one of its initial events then it takes over, defined corecursively.

$$(\mathit{Sil}\, P') \triangle Q = \mathit{Sil}\, (P' \triangle Q) \qquad P \triangle (\mathit{Sil}\, Q') = \mathit{Sil}\, (P \triangle Q')$$

$$(\mathit{Ret}\, x) \triangle Q = \mathit{Ret}\, x \qquad P \triangle (\mathit{Ret}\, x) = \mathit{Ret}\, x$$

$$(\mathit{Vis}\, F) \triangle (\mathit{Vis}\, G) = \mathit{Vis}\, \big(\{e \mapsto (P' \triangle Q) \mid (e \mapsto P') \in (\mathsf{dom}(G) \lhd F)\} \oplus G \big)$$

ROBOSTAR

Extra CSP operators - Interrupt

Definition (Interrupt)

$P \triangle Q$ behaves like P except that if at any time Q performs one of its initial events then it takes over, defined corecursively.

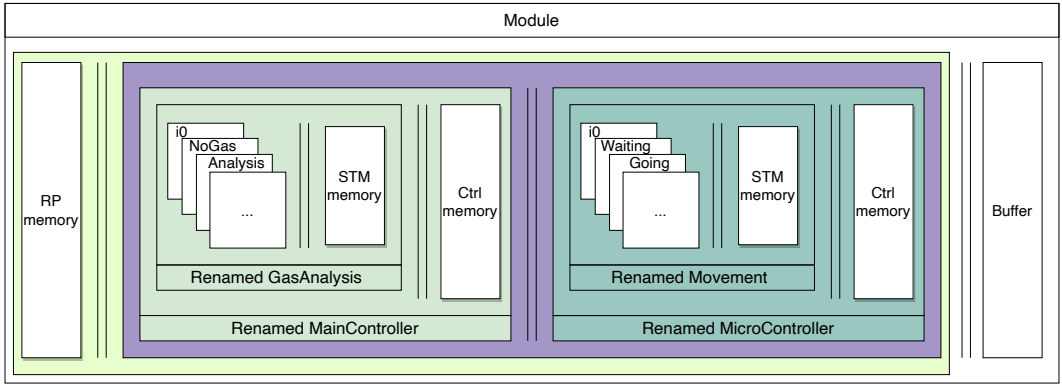
$$(\mathit{Sil}\, P') \triangle Q = \mathit{Sil}\, (P' \triangle Q) \qquad P \triangle (\mathit{Sil}\, Q') = \mathit{Sil}\, (P \triangle Q')$$
$$(\mathit{Ret}\, x) \triangle Q = \mathit{Ret}\, x \qquad P \triangle (\mathit{Ret}\, x) = \mathit{Ret}\, x$$
$$(\mathit{Vis}\, F) \triangle (\mathit{Vis}\, G) = \mathit{Vis}\, \big(\ \{e \mapsto (P' \triangle Q) \mid (e \mapsto P') \in (\mathsf{dom}(G) \lhd F)\} \oplus G\ \big)$$

Example

- $(\mathit{Ret}\, x) \triangle Q = \mathit{Ret}\, x$, and $(a \to P) \triangle (\mathit{Ret}\, x) = \mathit{Ret}\, x$
- Priority to Q: $(a \to P) \triangle (a \to Q) = a \to Q$

ROBOSTAR

RoboChart semantics



ROBOSTAR

RoboChart data types

RoboChart types	Isabelle types	Note
basic primitive types: nat, int, ...	`natural`, `integer`, ...	target language types
abstract primitive type	`PrimType`	Chem, Intensity
enumerations	**`datatype`**	
records	**`record`**	
mathematical types: sets, relations, ...	Z mathematical toolkit	Bounded finite sequences

Definition (Abstract primitive type to finite enumerations)

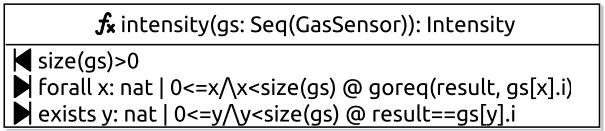
```
datatype ('t, 'a::finite) PrimType = PrimTypeC 'a
```

Definition (Bounded finite sequence)

```
typedef ('a,'n::finite) blist = {xs::'a list. length xs≤CARD('n)}
```

ROBOSTAR

RoboChart functions



Definition (intensity function)

```
definition "pre_Chemical_intensity gs = (blength gs > 0)"
definition "Chemical_intensity gs = (THE result.
  (∀x::nat<blength gs. Chemical_goreq(result, gs_i (bnth gs x)))∧
  (∃x::nat<blength gs. result = gs_i (bnth gs x)))"
```

Detected an error in the original model where x <= size(gs) is used.

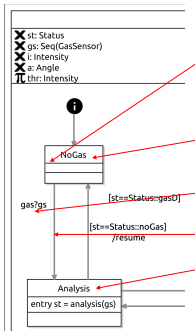
Channel type

A process is of type (E, V)*itree*;
Two processes in parallel composition have the same type: E and V.

```
chantype Chan_Movement =
  internal_Movement  :: TIDS_Movement
  terminate_Movement :: unit
  enter_Movement     :: "SIDS_Movement×SIDS_Movement" ...
  get_l_Movement     :: "Location_Loc"
  set_l_Movement     :: "Location_Loc"
  set_EXT_l_Movement :: "Location_Loc"
  obstacle__Movement :: "TIDS_Movement×InOut×Location_Loc"
  obstacle_Movement  :: "InOut×Location_Loc" ...
  moveCall_Movement  :: "core_real×Chemical_Angle" ...
```

ROBOSTAR

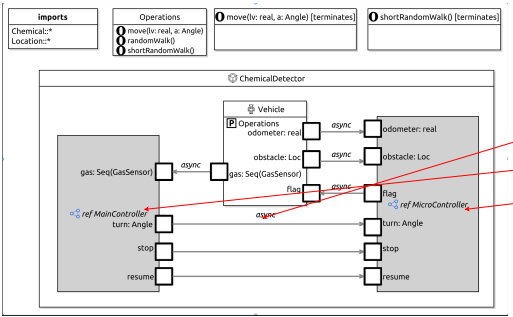
RoboChart States




```
definition State_GasAnalysis_NoGas where
"State_GasAnalysis_NoGas =
  loop (λ (id::integer).
    do {sd ← inp_in enter_GasAnalysis_Other_SIDs_to_NoGas_GasAnalysis ;
        — ‹ State passed to next loop, including a condition initially True. ›
        ret ← Ret (True, id, fst sd) ;
        — ‹ @{verbatim State_GasAnalysis_NoGas_execute} ›
        (iterate
           — ‹ condition ›
           (λ s. fst s)
           — ‹ P ›
           (λ s.
            do {
            outp_entered_GasAnalysis (snd (snd s), SID_GasAnalysis_NoGas);
            (do {skip ; stop} △
              (
              — ‹ @{verbatim T_GasAnalysis_t2} ›
              do {t ← inp_in gas__GasAnalysis (set [(TID_GasAnalysis_t2, din, gs)
                                 gs ← (lseq gassensor_enum)]) ;
                  outp_set_gs_GasAnalysis (snd (snd t)) ;
                  outp_exit_GasAnalysis (SID_GasAnalysis_NoGas, SID_GasAnalysis_NoGas);
                  outp_exited_GasAnalysis (SID_GasAnalysis_NoGas, SID_GasAnalysis_NoGas);
                  outp_enter_GasAnalysis (SID_GasAnalysis_NoGas, SID_GasAnalysis_Analysis);
                  outp_entered_GasAnalysis (SID_GasAnalysis_NoGas, SID_GasAnalysis_Analysis);
                  Ret(False, fst (snd s), SID_GasAnalysis_NoGas)
                 } □
              (exit_events_GasAnalysis (fst (snd s)) SID_GasAnalysis_NoGas
                tids_GasAnalysis_NoGas Other_SIDs_to_NoGas_GasAnalysis)
              )
            )
           })
           — ‹ The previous state: a triple ›
           (ret)
        ) ;
        Ret (id)
    }
  )
"
```

ROBOSTAR

RoboChart Module




```
definition D__ChemicalDetector where
"D__ChemicalDetector (idd::integer) =
  (
    (
      (par_hide
        (discard_state (buffer0 []))
        ChemicalDetector_controllers_buffer_events
        (
          (
            (
              (rename_D__MainController idd)
              ||ₛ (ChemicalDetector_controllers_sync_events) ↙
              (rename_D__MicroController idd)
            ) \ (ChemicalDetector_controllers_sync_events - (set [terminate_C ()]))
          )
          ||ₛ (set []) ↙
          Memory_ChemicalDetector
        )
      ) ⟦ set [terminate_C ()] ▷ skip
    ) \ (set [terminate_C ()])
  )
"
```

ROBOSTAR

Conclusion

- Operational semantics for RoboChart, ITrees-based CSP,

ROBOSTAR

Conclusion

- Operational semantics for RoboChart, ITrees-based CSP,
- Manual generation of the semantics of the autonomous chemical detector model,

ROBOSTAR

Conclusion

- Operational semantics for RoboChart, ITrees-based CSP,
- Manual generation of the semantics of the autonomous chemical detector model,
- Animation of the model,

ROBOSTAR

Conclusion

- Operational semantics for RoboChart, ITrees-based CSP,
- Manual generation of the semantics of the autonomous chemical detector model,
- Animation of the model,
- Implementation of three extra CSP operators and one bounded finite sequence type.

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,
- Graphical animation (in RoboChart state machines),

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,
- Graphical animation (in RoboChart state machines),
- Support more RoboChart constructs: composite states, timed semantics,

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,
- Graphical animation (in RoboChart state machines),
- Support more RoboChart constructs: composite states, timed semantics,
- Verification (refinement or temporal logics) in addition to animation,

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,
- Graphical animation (in RoboChart state machines),
- Support more RoboChart constructs: composite states, timed semantics,
- Verification (refinement or temporal logics) in addition to animation,
- Extension to other semantics domains: probability (DTMC), ...

ROBOSTAR

Future work

- Nondeterminism: static (resolved in the beginning), oracle (another process to resolve), or randomly,
- Automation of semantics generation,
- Graphical animation (in RoboChart state machines),
- Support more RoboChart constructs: composite states, timed semantics,
- Verification (refinement or temporal logics) in addition to animation,
- Extension to other semantics domains: probability (DTMC), ...
- Applications: sound runtime monitors, concrete implementation of RoboChart controllers (for verified ROS nodes).

ROBOSTAR

References

- Kazmierczak et al. Verifying model oriented specifications through animation. APSEC. (1998).
- Hilder et al. Chemical detection using the receptor density algorithm. IEEE Trans. Syst. Man Cybern. (2012)
- Miyazawa et al. RoboChart: modelling and verification of the functional behaviour of robotic applications. Softw. Syst. Model. (2019)
- Xia et al. Interaction trees: representing recursive and impure programs in Coq. POPL (2019)
- Foster et al. Formally verified simulations of state-rich processes using interaction trees in Isabelle/HOL. CONCUR (2021)

ROBOSTAR

Thank you!

https://robostar.cs.york.ac.uk/

ROBOSTAR