Complete Test of Synthesised Safety Supervisors

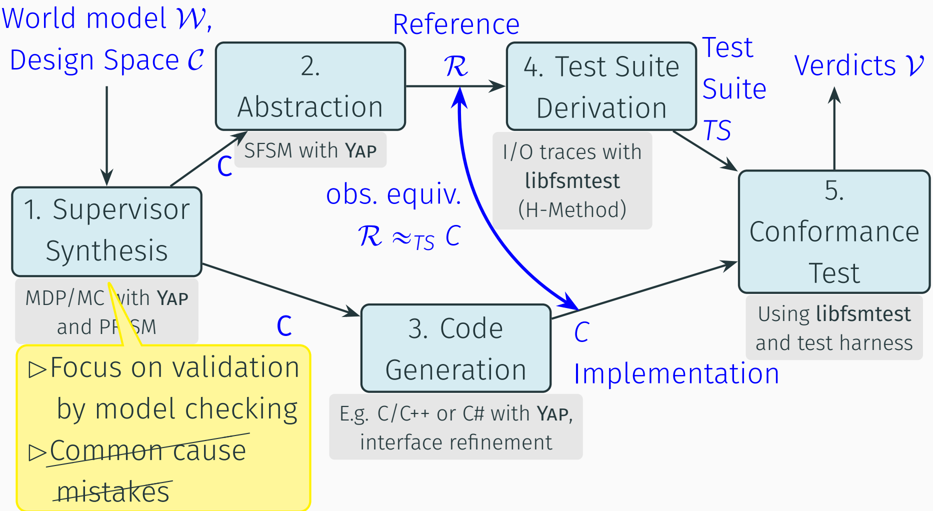
Mario Gleirscher[1,2] and Jan Peleska[1]

FMAS, 3rd Workshop, 22 October 2021

[1]University of Bremen, Germany
[2]Autonomy Assurance International Programme, York, UK

Open Educational Resource License Agreement

Synthesis-informed Conformance Test of Supervisors



World model $\mathcal{W}$, Design Space $\mathcal{C}$

1. Supervisor Synthesis

MDP/MC with Yap and PRISM

C

2. Abstraction

SFSM with Yap

Reference $\mathcal{R}$

obs. equiv. $\mathcal{R} \approx_{TS} C$

4. Test Suite Derivation

I/O traces with **libfsmtest** (H-Method)

Test Suite TS

C

3. Code Generation

E.g. C/C++ or C# with Yap, interface refinement

C Implementation

5. Conformance Test

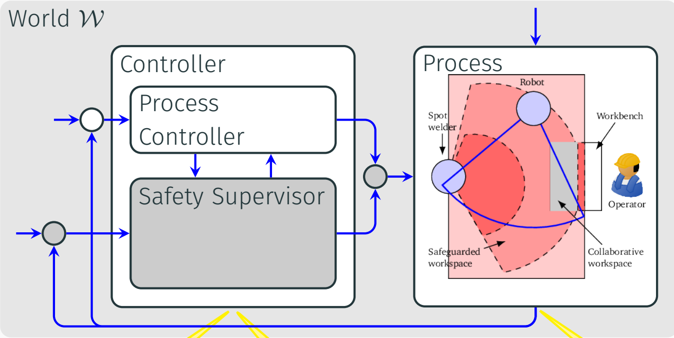
Using **libfsmtest** and test harness

Verdicts $\mathcal{V}$

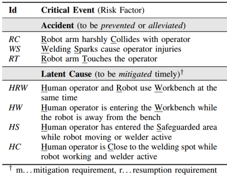
- ▷ Focus on validation by model checking
- ▷ ~~Common cause mistakes~~

Mario Gleirscher and Jan Peleska (2021). “Complete Test of Synthesised Safety Supervisors for Robots and Autonomous Systems”. In: *Formal Methods for Autonomous Systems (FMAS), 3rd Workshop*. Vol. 348

HRC Safety: Verified Supervisor Synthesis & Test



Hazards (specs)



Id	**Critical Event** (Risk Factor)
	Accident (to be *prevented* or *alleviated*)
RC	Robot arm harshly Collides with operator
WS	Welding Sparks cause operator injuries
RT	Robot arm Touches the operator
	Latent Cause (to be *mitigated* timely)†
HRW	Human operator and Robot use Workbench at the same time
HW	Human operator is entering the Workbench while the robot is away from the bench
HS	Human operator has entered the Safeguarded area while robot moving or welder active
HC	Human operator is Close to the welding spot while robot working and welder active

† m. . . mitigation requirement, r. . . resumption requirement

Functional Safety
avoid harm
by meeting specs

+

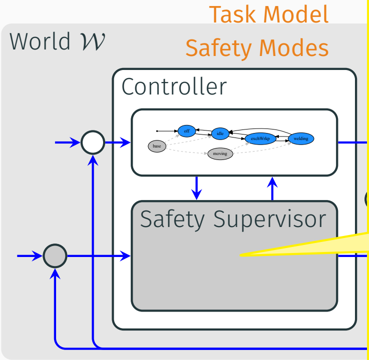
SOTIF[1]
avoid harm
by safe behaviour

⟹

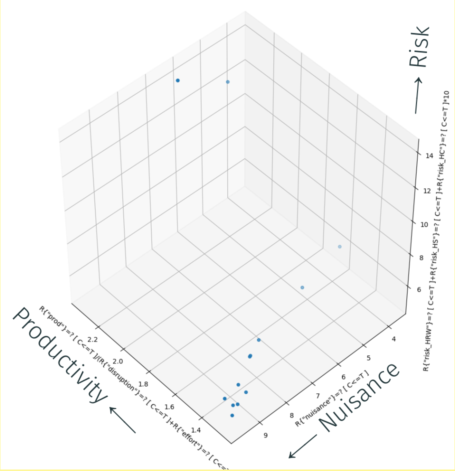
Operational Safety
avoid harm
by reducing hazards

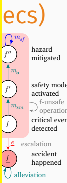
[1] HRC: human-robot collaboration, SOTIF: safety of the intended function

HRC Safety: Verified Supervisor Synthesis & Test

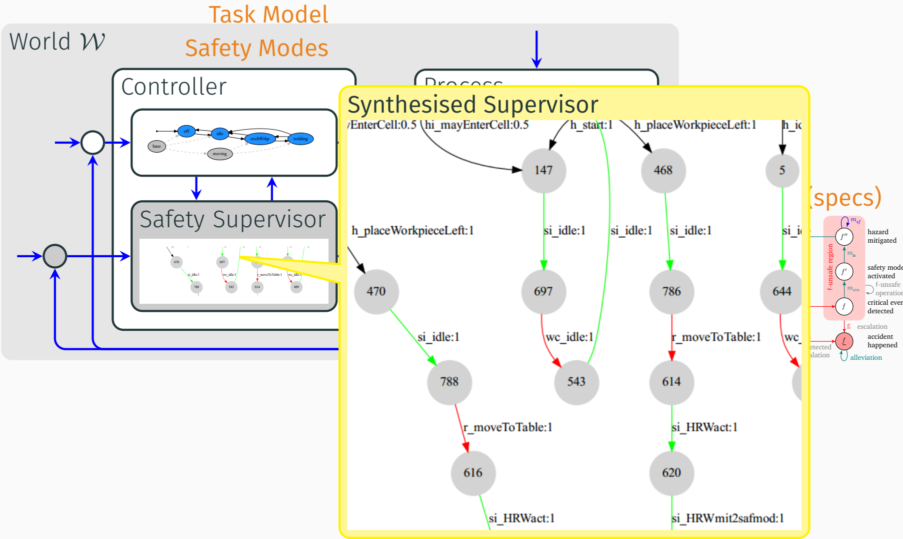


Optimal synthesis from design space $\mathcal{C}$

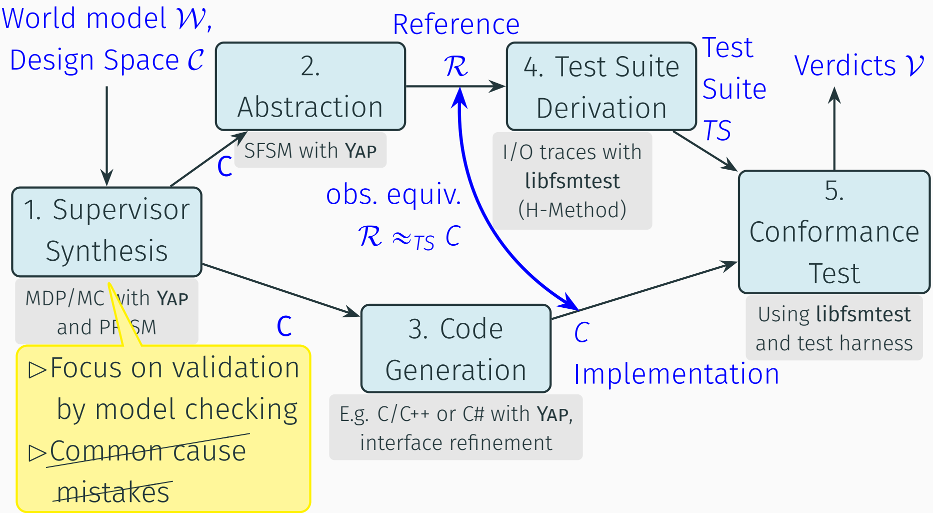




HRC Safety: Verified Supervisor Synthesis & Test



Synthesis-informed Conformance Test of Supervisors



World model $\mathcal{W}$, Design Space $\mathcal{C}$

1. Supervisor Synthesis

MDP/MC with YAP and PRISM

C

2. Abstraction

SFSM with YAP

Reference $\mathcal{R}$

obs. equiv. $\mathcal{R} \approx_{TS} C$

4. Test Suite Derivation

I/O traces with **libfsmtest** (H-Method)

Test Suite TS

3. Code Generation

E.g. C/C++ or C# with YAP, interface refinement

C

C Implementation

5. Conformance Test

Using **libfsmtest** and test harness

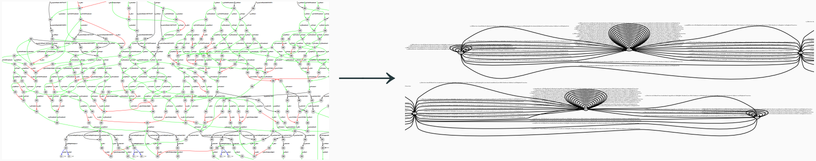
Verdicts $\mathcal{V}$

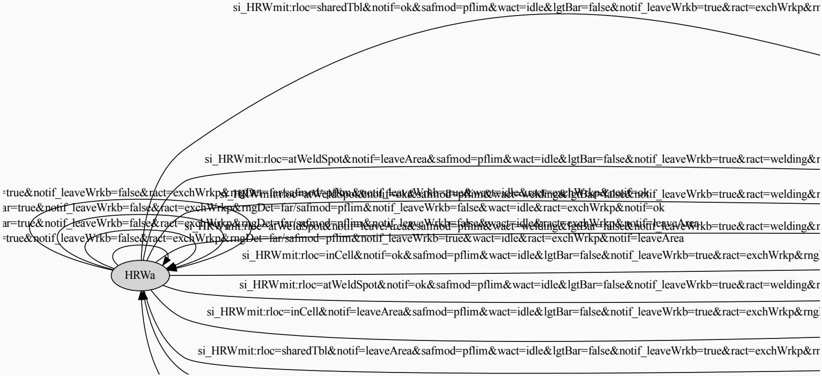
▷ Focus on validation by model checking

▷ ~~Common cause mistakes~~

Mario Gleirscher and Jan Peleska (2021). “Complete Test of Synthesised Safety Supervisors for Robots and Autonomous Systems”. In: *Formal Methods for Autonomous Systems (FMAS), 3rd Workshop*. Vol. 348

Test Reference $\mathcal{R}$ (the oracle)




si_HRWmit:rloc=sharedTbl&notif=ok&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=exchWrkp&rr
si_HRWmit:rloc=atWeldSpot&notif=leaveArea&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=welding&r
ar=true&notif_leaveWrkb=false&ract=exchWrkp&rngDet=far/safmod=pflim&notif_leaveWrkb=false&wact=idle&ract=exchWrkp&notif=ok
=true&notif_leaveWrkb=false&ract=exchWrkp&rngDet=far/safmod=pflim&notif_leaveWrkb=true&wact=idle&ract=exchWrkp&notif=leaveArea
si_HRWmit:rloc=inCell&notif=ok&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=exchWrkp&rng
HRWa
si_HRWmit:rloc=atWeldSpot&notif=ok&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=welding&r
si_HRWmit:rloc=inCell&notif=leaveArea&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=exchWrkp&rng
si_HRWmit:rloc=sharedTbl&notif=leaveArea&safmod=pflim&wact=idle&lgtBar=false&notif_leaveWrkb=true&ract=exchWrkp&rr

Example of a C++-based Controller Implementation C

```
...
void sampleAndControl() { // scenarios are disjoint
  if (evalGuard58()) {
    printf("\nTriggered guard 58");
    controlAndRiskUpdate58();
  }
  ...
}
...
bool evalGuard58() {
  return rloc == SHAREDTBL && notif == OK && safmod == PFLIM
    && wact == IDLE && lgtBar == true && notif_leaveWrkb == false
    && ract == EXCHWRKP && rngDet == FAR;
}
...
void controlAndRiskUpdate58() { // only controller vars and risk state
  if (HSp == INACT && HCp == INACT && HRWp == ACT) { // si_HRWmit2fun
    notif_leaveWrkb = true;
    printf("\nHandler changed notif_leaveWrkb");
  } else {
    isNull = true;
  }
}
...
```

Input Sequences of test cases A, B, and C

Sequence A with 2 inputs:

```
A1. rloc=inCell&notif=ok&safmod=pflim&wact=idle&...,
A2. rloc=atWeldSpot&... ract=welding&rngDet=far
```

...B with 2 inputs:

```
B1. rloc=inCell&...&notif_leaveWrkb=false&ract=exchWrkp&...,
B2. rloc=sharedTbl&notif=ok&...&ract=exchWrkp&rngDet=near
```

...C with 3 inputs:

```
C1. rloc=sharedTbl&notif=ok&safmod=normal&...&rngDet=far,
C2. rloc=inCell&notif=leaveArea&safmod=normal&...&rngDet=far,
C3. rloc=inCell&notif=leaveArea&...&ract=exchWrkp&rngDet=far
```

Positive Verdict $\in \mathcal{V}$ for a test case with 4 I/O pairs

Test case:

```
rloc=sharedTbl&... / safmod=normal&...,
rloc=sharedTbl&... / safmod=pflim&...&notif=leaveArea,
rloc=inCell&...&rngDet=far / safmod=normal&...,
rloc=inCell&...&rngDet=far / null
```

Test protocol:

```
Triggered guard 56, Handler changed /HRWp
Output 0: sharedTbl,ok,normal,idle,true,false,exchWrkp,far,
Triggered guard 9, Handler changed /HRWp
Output 1: sharedTbl,leaveArea,pflim,idle,false,true,exchWrkp,far,
Triggered guard 5, Handler changed /HRWp
Output 2: inCell,leaveArea,normal,idle,false,false,exchWrkp,far,
Triggered guard 5
Output 3: inCell,leaveArea,normal,idle,false,false,exchWrkp,far,
```

Verdict: PASS

Threats to Validity (and Their Refutation)

Core assumptions for guaranteed fault coverage by *TS*:

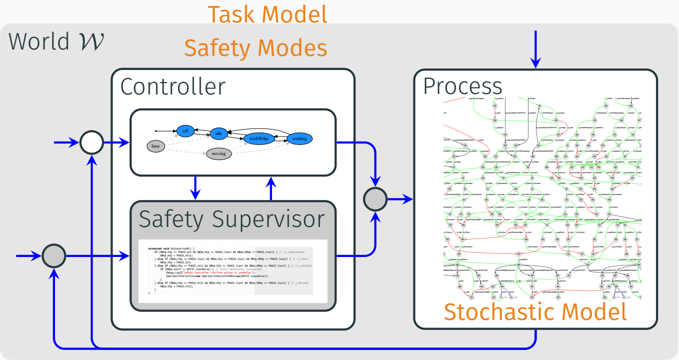
- $|\mathcal{R}| \geqslant |C|$ ... (at most) as many control states in C
- $\mathcal{R}$ is deterministic, incl. isomorphism from $[I_C^*]_\approx$ to $[I_\mathcal{R}^*]_\approx$

Error possibilities:

- Error in the generation of $\mathcal{R}$ $\Longrightarrow$ verified generator
- Error in the testing theory $\Longrightarrow$ mechanised theory
- H-method implementation error $\Longrightarrow$ automatic check
- Test harness error $\Longrightarrow$ mutation testing

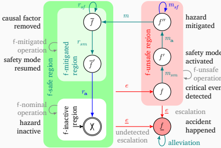
Future work: Formal refutation of error possibilities

HRC Safety: Verified Supervisor Synthesis & Test



```
protected void OnScenario40() {
    if (this.HSp == PHASE.act && this.HCp == PHASE.inact && this.HRWp == PHASE.inact) { // si_ssmonsofmod
        this.HSp = PHASE.mit1;
    } else if (this.HSp == PHASE.inact && this.HCp == PHASE.inact && this.HRWp == PHASE.inact) { // si_HSact
        this.HSp = PHASE.act;
    } else if (this.HSp == PHASE.mit2 && this.HCp == PHASE.inact && this.HRWp == PHASE.inact) { // si_ssmonfun
        if (this.notif != NOTIF.leaveArea) { // avoid idempotency assumption
            Debug.Log($"Safety Controller::Perform action si_ssmonfun");
            OperatorInteraction(new OperatorInteractionMessage(NOTIF.leaveArea));
        }
    } else if (this.HSp == PHASE.mit1 && this.HCp == PHASE.inact && this.HRWp == PHASE.inact) { // s_HSssmon
        this.HSp = PHASE.mit2;
    }
}
```

Risk Model (specs)



Next Steps:

▷ System test theory

▷ Increased robustness

▷ Appl. to mobile robots

Gleirscher/Peleska, *FMAS* (vol. 348) 2021

Gleirscher/Calinescu/Douthwaite, *CoRR* 2021

Gleirscher/Calinescu/Woodcock, *FAOC* (33) 2021

Demo in YAP 0.8+, yap.gleirscher.de ⇒ Artifacts Evaluated Functional acm ?

Gleirscher, Mario, Radu Calinescu, James Douthwaite, et al. (2021). *Verified Synthesis of Optimal Safety Controllers for Human-Robot Collaboration*. Working paper arXiv:abs/2106.06604. U York, U Sheffield, et al. arXiv: 2106.06604 [cs.RO cs.SE].

Gleirscher, Mario, Radu Calinescu, and Jim Woodcock (2021). “Risk Structures: A Design Algebra for Risk-Aware Machines”. In: *Form Asp Comput* 33, pp. 763–802. arXiv: 1904.10386 [cs.SE].

Gleirscher, Mario and Jan Peleska (2021). “Complete Test of Synthesised Safety Supervisors for Robots and Autonomous Systems”. In: *Formal Methods for Autonomous Systems (FMAS), 3rd Workshop*. Vol. 348. EPTCS, pp. 101–109.