WWU
MÜNSTER

Asynchronous Hyperproperties

Our current work on the asynchronous hyperlogic mumbling H_μ

Christoph Ohrem

wissen.leben

AG Softwareentwicklung und
Verifikation
Institut für Informatik

Verification by Model Checking

Model Checking problem:
Given a model $\mathcal{M}$ and a property φ, check whether

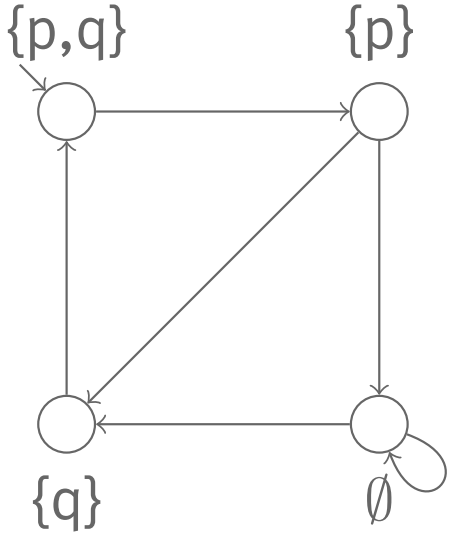
$$\mathcal{M} \models \varphi$$

Model of a program, hardware circuit, ...

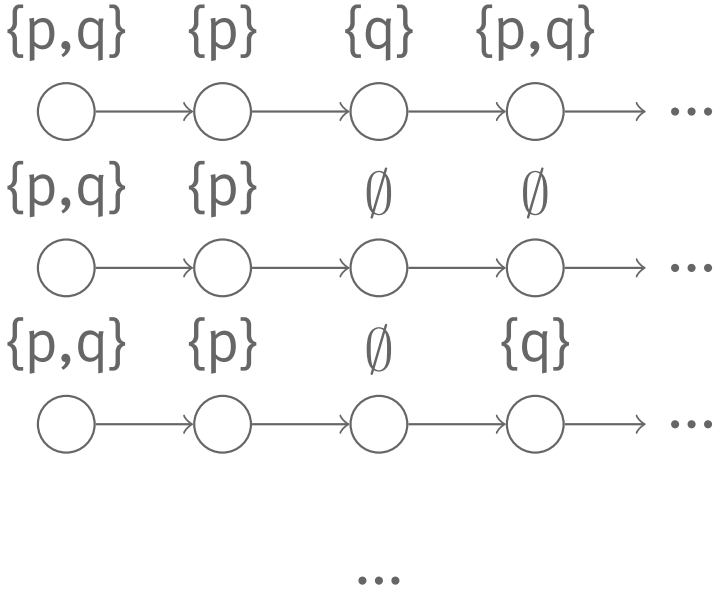
Desirable/undesirable property

Models - Kripke Structures

Kipke Structure $\mathcal{K}$



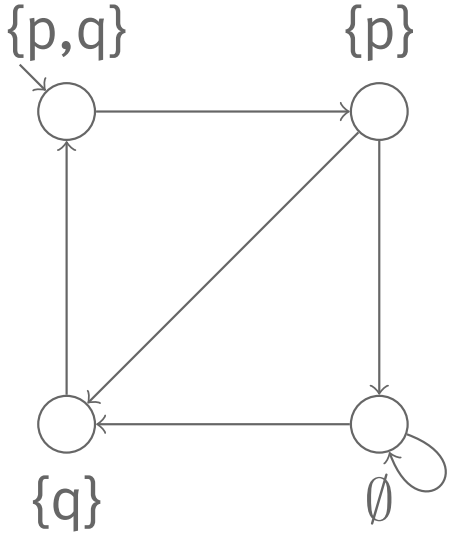
Paths



p,q: atomic propositions from a set *AP*

Models - Kripke Structures

Kipke Structure $\mathcal{K}$



Traces

{p,q} {p} {q} {p,q} ⋯

{p,q} {p} ∅ ∅ ⋯

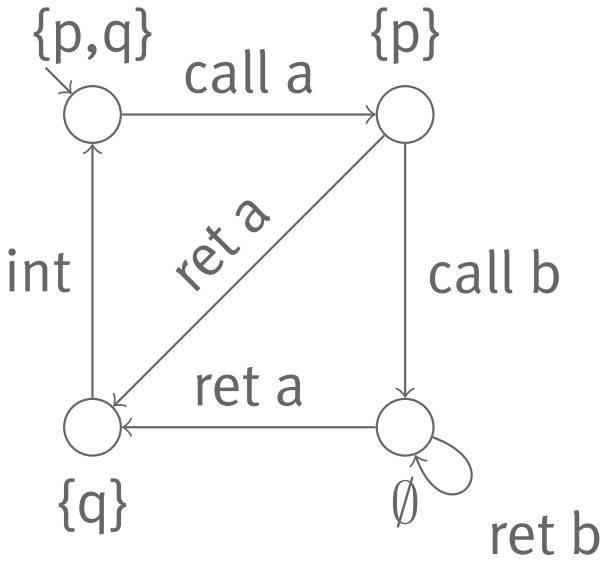
{p,q} {p} ∅ {q} ⋯

⋯

p,q: atomic propositions from a set *AP*

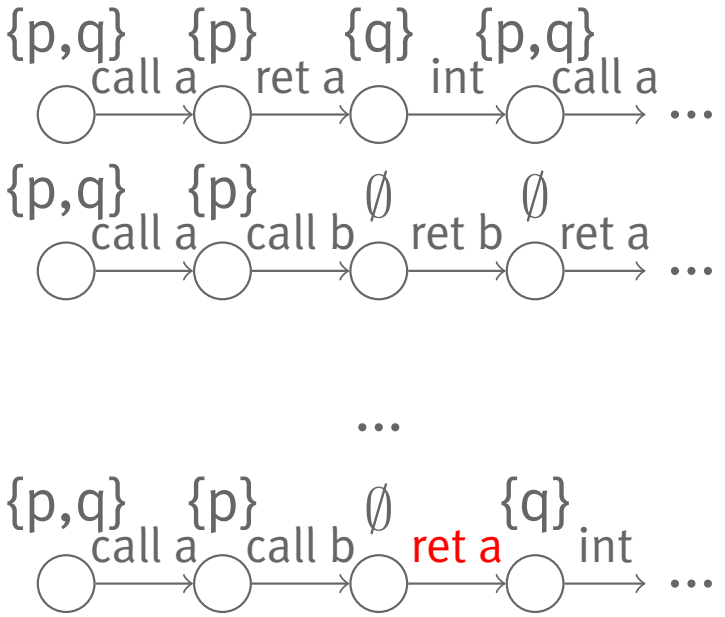
Models - Pushdown Systems

Pushdown System $\mathcal{P}$



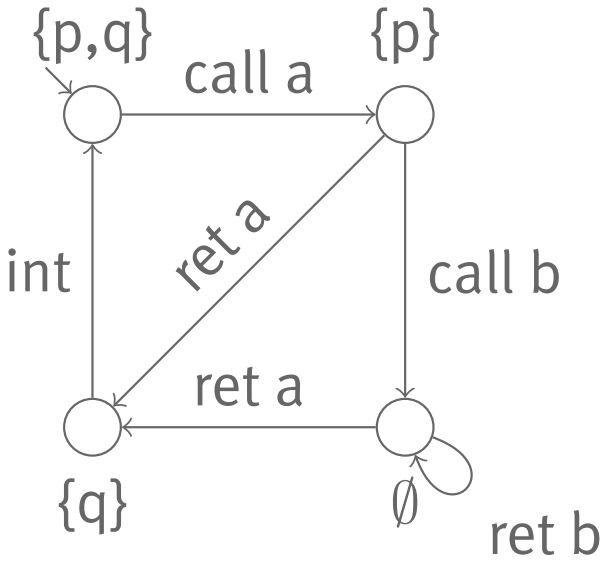
a,b: stack symbols from a set Θ

Paths



Models - Pushdown Systems

Pushdown System $\mathcal{P}$



a,b: stack symbols from a set Θ

Traces

{p,q}c {p} r {q} i {p,q}c ⋯

{p,q}c {p} c ∅ r ∅ r ⋯

⋯

Properties - Linear Temporal Logics (LTL)[1]

"p or q always holds"

$\mathcal{G}(p \vee q)$

{p,q} {p} {q} {p,q} ··· ✓

{p,q} {p} ∅ {q} ··· ✗

LTL formulae are satisfied by traces; Properties are sets of traces

[1] [Pnueli, SFCS 1977]

Properties - CaRet[2]

"q holds when the currently called procedure is returned from"

$$\bigcirc^a q$$

(assuming the next transition is a call)

{p,q}	c	{p}	r	{q}	i	{p,q}	c	⋯	✓	
{p,q}	c	{p}	c	∅	r	∅	r	{q}	⋯	✓

"procedure f is in the call stack"

$$\mathcal{F}^c f$$

(assuming each state is labelled with the current top of stack symbol)

[2] [Alur et al., TACAS 2004]

Model Checking LTL

Main Idea for Automata based LTL Model Checking

Kripke Structure $\mathcal{K}$ $\xrightarrow[\text{translation}]{\text{linear}}$ NBA[3] $\mathcal{A}_{\mathcal{K}}$
$\mathcal{L}(\mathcal{A}_{\mathcal{K}}) = \mathit{Traces}(\mathcal{K})$

LTL Formula φ $\xrightarrow[\text{translation}]{\text{linear}}$ ABA $\mathcal{A}_{\neg\varphi}$ $\xrightarrow[\text{dealternation}]{\text{exponential}}$ NBA $\mathcal{A}_{\neg\varphi}$
$\mathcal{L}(\mathcal{A}_{\neg\varphi}) = \{tr \mid tr \not\models \varphi\}$

$\mathcal{K} \models \varphi$ iff $\mathcal{L}(\mathcal{A}_{\mathcal{K}}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$; Model Checking is in PSPACE

[3] NBA = Nondeterministic Büchi Automaton, ABA = Alternating Büchi Automaton

Synchronous Hyperproperties - HyperLTL[4]

"The system seems to be deterministic to a low security user"

$$\varphi_{OD} = \forall \pi_1 . \forall \pi_2 . (i_{\pi_1} \leftrightarrow i_{\pi_2}) \rightarrow \mathcal{G}(o_{\pi_1} \leftrightarrow o_{\pi_2})$$

tr_1:	{i}	{o}	{h}	{o}	⋯
tr_2:	{i}	{o}	∅	{o,h}	⋯
tr_3:	{i}	{h}	{o}	{o}	⋯
tr_4:	∅	{h}	{o}	{o}	⋯

$\{tr_1, tr_2, tr_4\}$ ✓

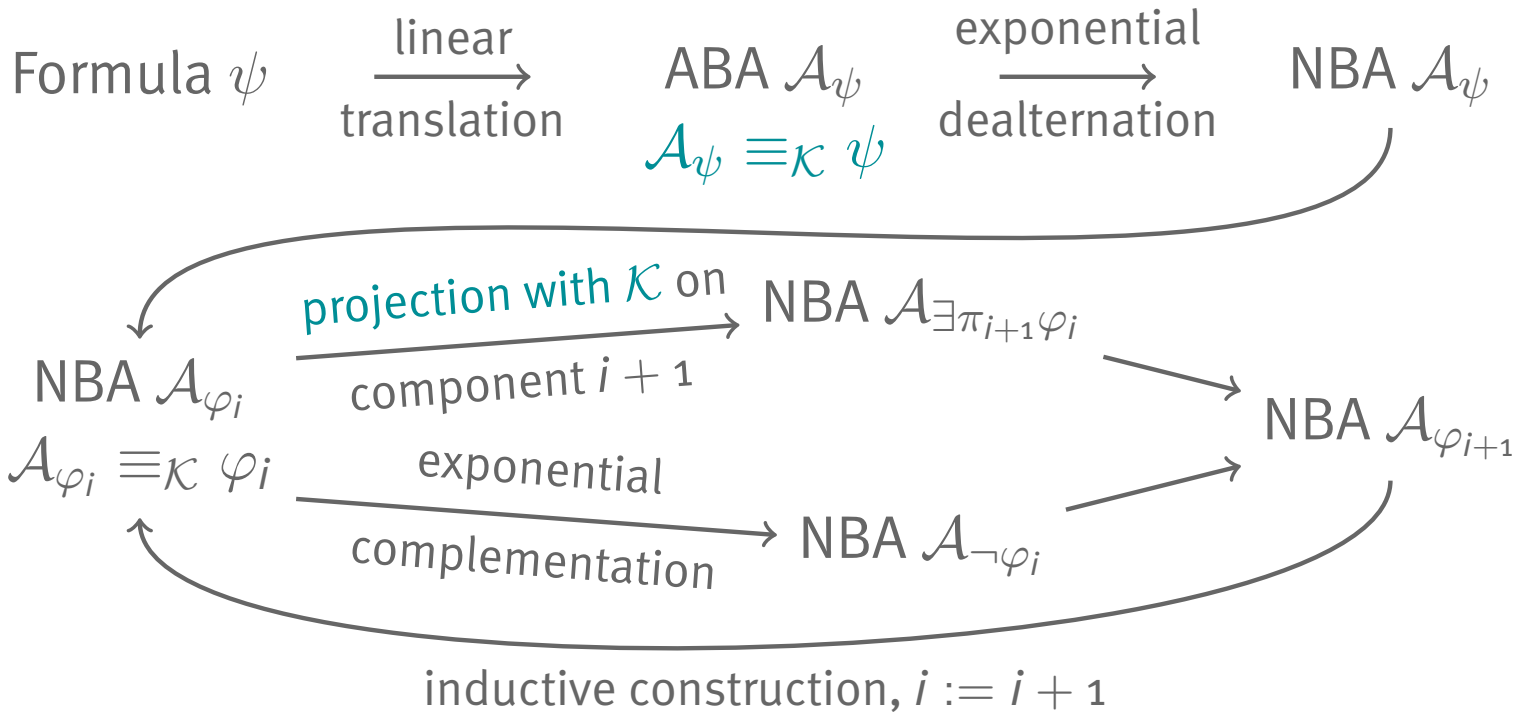
$\{tr_1, tr_3\}$ ✗

Inner HyperLTL formulae are satisfied by trace assignments;
Closed HyperLTL formulae are satisfied by sets of traces;
Hyperproperties are sets of sets of traces

[4] [Clarkcon et al., POST 2014], Model Checking in [Finkbeiner et al., CAV 2015]

Model Checking HyperLTL

Formula $\varphi = Q_n \pi_n \dots Q_1 \pi_1 . \psi$ with $\varphi_i = Q_i \pi_i \dots Q_1 \pi_1 . \psi$, Structure $\mathcal{K}$



$\mathcal{K} \models \varphi$ iff $\mathcal{L}(\mathcal{A}_\varphi) \neq \emptyset$; Model Checking is in *k*EXPSPACE

Asynchronous Hyperproperties - H_μ[5]

Asynchronicity arises naturally in many programs:

```
while (true) {
        a = a + c;
}
```

```
while (true) {
        temp = a + c;
        a = temp;
}
```

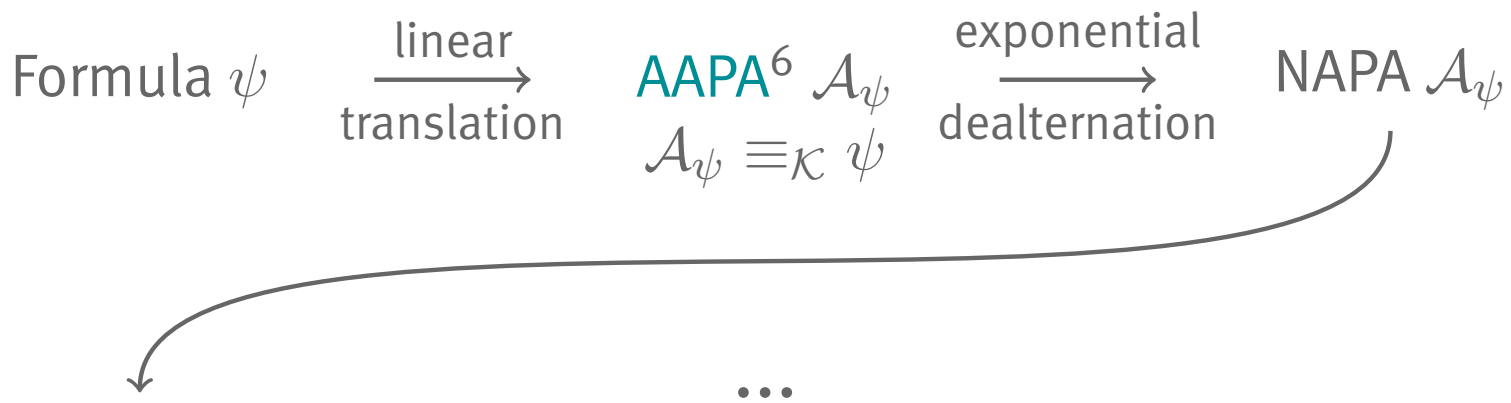
H_μ generally captures asynchronous hyperproperties like "The traces π_1 and π_2 repeatedly agree on a, but one step on π_1 is matched by two on π_2"

$$\nu X.((a_{\pi_1} \leftrightarrow a_{\pi_2}) \wedge \bigcirc_{\pi_1} \bigcirc_{\pi_2} \bigcirc_{\pi_2} X)$$

[5] [Gutsfeld et al., POPL 2021]

Model Checking H_μ

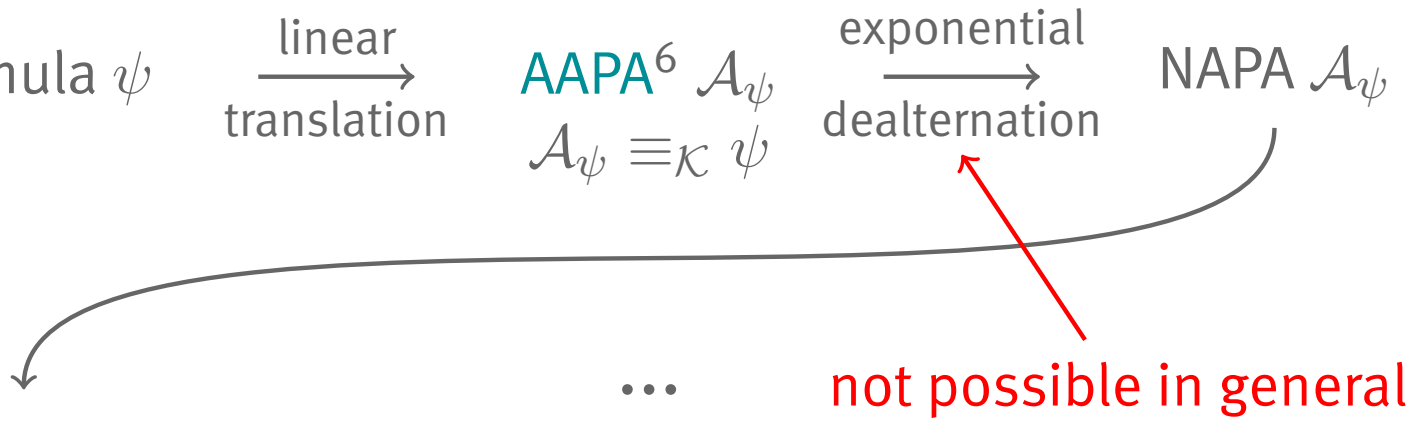
Formula $\varphi = Q_n \pi_n \ldots Q_1 \pi_1 . \psi$ with $\varphi_i = Q_i \pi_i \ldots Q_1 \pi_1 . \psi$, Structure $\mathcal{K}$



[6] Alternating Asynchronous Parity Automaton [Gutsfeld et al., POPL 2021]

Model Checking H_μ

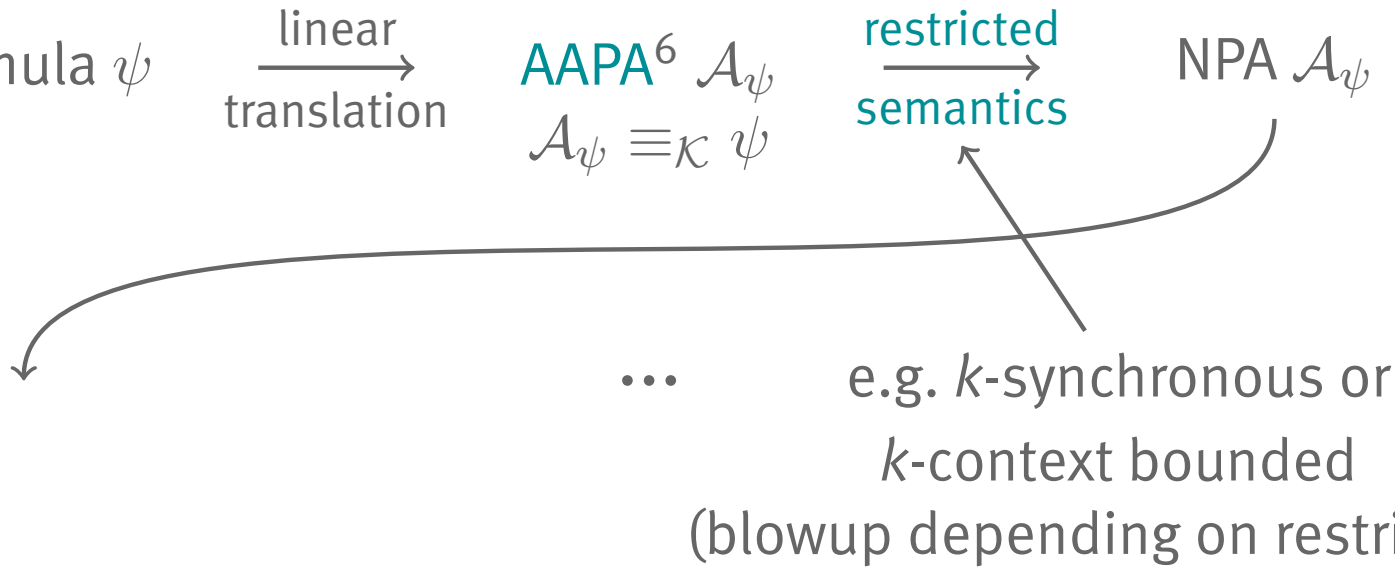
Formula $\varphi = Q_n \pi_n \dots Q_1 \pi_1 . \psi$ with $\varphi_i = Q_i \pi_i \dots Q_1 \pi_1 . \psi$, Structure $\mathcal{K}$



[6] Alternating Asynchronous Parity Automaton [Gutsfeld et al., POPL 2021]

Model Checking H_μ

Formula $\varphi = Q_n \pi_n \ldots Q_1 \pi_1 . \psi$ with $\varphi_i = Q_i \pi_i \ldots Q_1 \pi_1 . \psi$, Structure $\mathcal{K}$

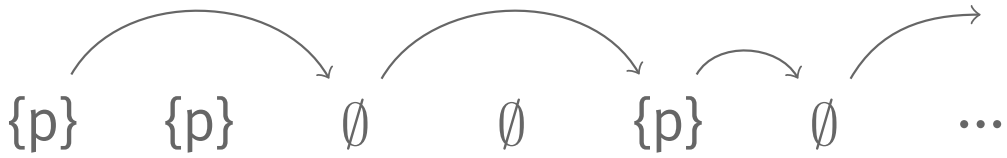


[6] Alternating Asynchronous Parity Automaton [Gutsfeld et al., POPL 2021]

Asynchronous Hyperproperties - Stuttering HyperLTL[7]

Define a stuttering criterion, inspect only one copy of the suttered states

Stuttering with $\Gamma = \{p\}$:



Captures a specific kind of asynchronicity, subsumed by H_μ

Observational determinism with equivalence up to stuttering:

$$\forall \pi_1 . \forall \pi_2 . \bigwedge_{p \in I} (p_{\pi_1} \leftrightarrow p_{\pi_2}) \rightarrow \mathcal{G}^L \bigwedge_{p \in O} (p_{\pi_1} \leftrightarrow p_{\pi_2})$$

Decidable Model Checking for a very interesting subset of the logic:
Use only one stuttering criterion Γ

[7] [Bozzelli et al., LICS 2021]

(Fair) Model Checking Stuttering HyperLTL

Formula φ with single stuttering criterion Γ, fair Structure $(\mathcal{K}, F)$

$$\underset{(\mathcal{K},F)}{\varphi} \xrightarrow[\text{formulae from } \Gamma]{\text{remove non-atomic}} \underset{(\mathcal{K}',F')}{\varphi_{AP}} \xrightarrow[\text{transformation}]{\text{structure}} \underset{\mathcal{K}'_\Gamma}{\varphi^S_{AP}}$$

synchronous HyperLTL Model Checking

Model Checking still in *k*EXPSPACE

Asynchronous Hyperproperties - Mumbling H_μ[8]

Inspired by stuttering HyperLTL, but brings several additions:

- fixpoints
- non-atomic tests
- CaRet like modalities
- simpler jump mechanism
- different jump criteria for different traces

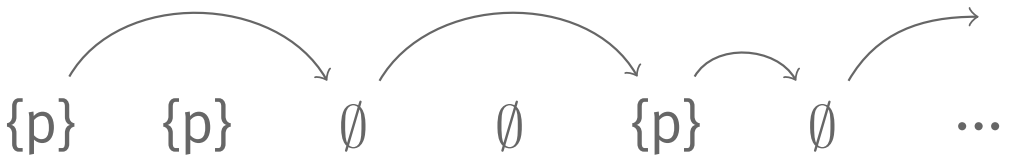
"The traces of the system are indistinguishable with respect to f being in the call stack"

$$\forall \pi_1.\forall \pi_2.\nu X.(([\mathcal{F}^c f]_{\pi_1} \leftrightarrow [\mathcal{F}^c f]_{\pi_2}) \wedge \bigcirc^{\Delta:\pi_1 \mapsto obs, \pi_2 \mapsto obs} X)$$

[8] Current work under review at POPL 2023

Stuttering vs. Mumbling

Stuttering with $\Gamma = \{p\}$:



Mumbling with $\Delta(\pi) = p$:

Mumbling seems more expressive: “all traces have the same number of occurrences of p” is expressible with an atomic mumbling criterion but not expressible with an LTL stuttering criterion

(Fair) Finite State Model Checking Mumbling H_μ

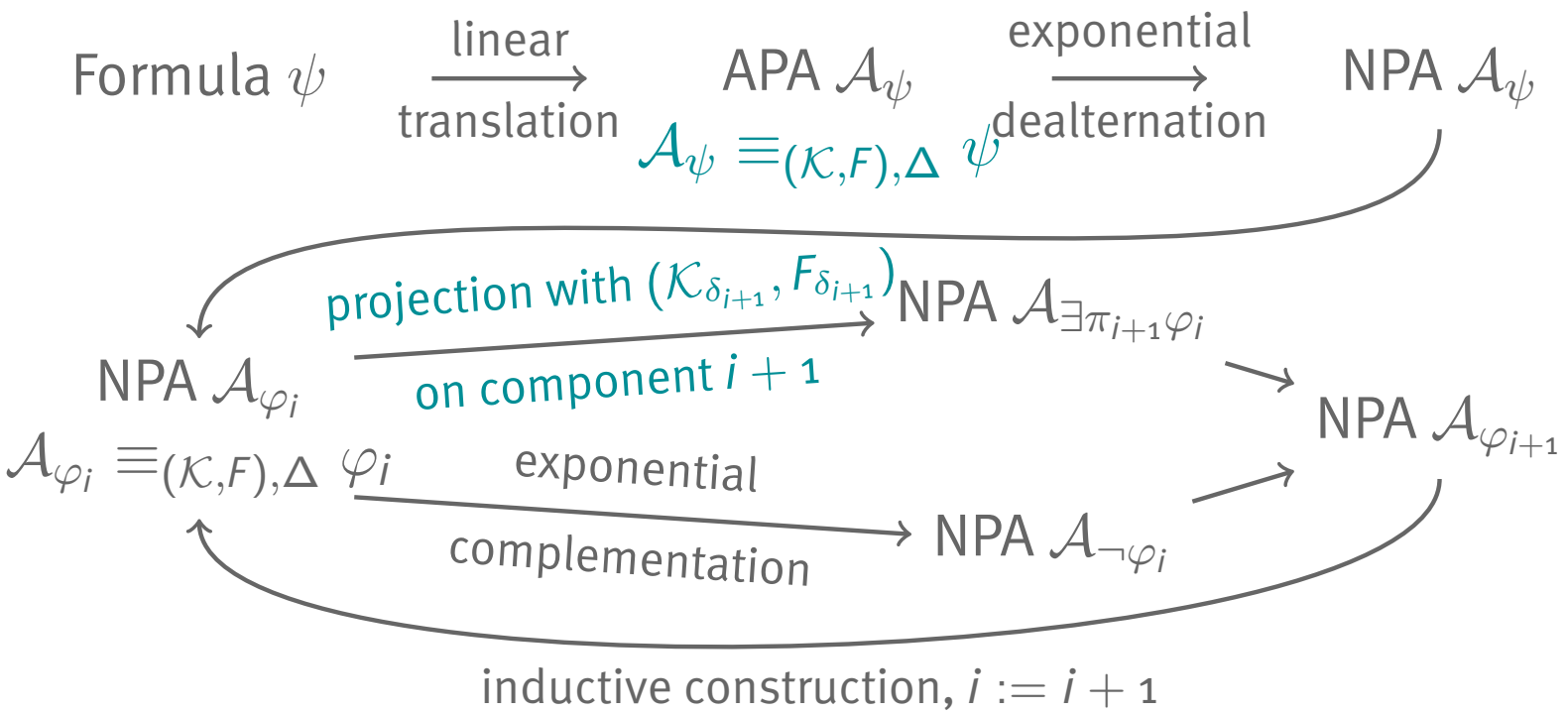
Formula φ with single mumbling criterion Δ, fair Structure $(\mathcal{K}, F)$

$$(\mathcal{K}, F)^{\varphi} \xrightarrow[\text{formulae from } \Delta \text{ and tests}]{\text{remove non-atomic}} (\mathcal{K}', F')^{\varphi_{AP}} \longrightarrow \text{Model Checking with basis } AP$$

Idea is very similar to stuttering HyperLTL Model Checking
Works for non-atomic tests, Pushdown Model Checking and CaRet-like modalities as well

(Fair) Finite State Model Checking Mumbling H_μ

Fair Model Checking with basis *AP*:



$\mathcal{K} \models \varphi$ iff $\mathcal{L}(\mathcal{A}_\varphi) \neq \emptyset$; Model Checking still in *k*EXPSPACE

Pushdown Model Checking - Undecidability

Reduction from emptiness problem for the intersection of contextfree languages; input model contains the two pushdown automata
Formula for the reduction:

$$\varphi = \exists \pi_1 . \exists \pi_2 . [a_1]_{\pi_1} \wedge [a_2]_{\pi_2} \wedge \mathcal{G} \bigwedge_{\sigma \in \Sigma} [\sigma]_{\pi_1} \leftrightarrow [\sigma]_{\pi_2}$$

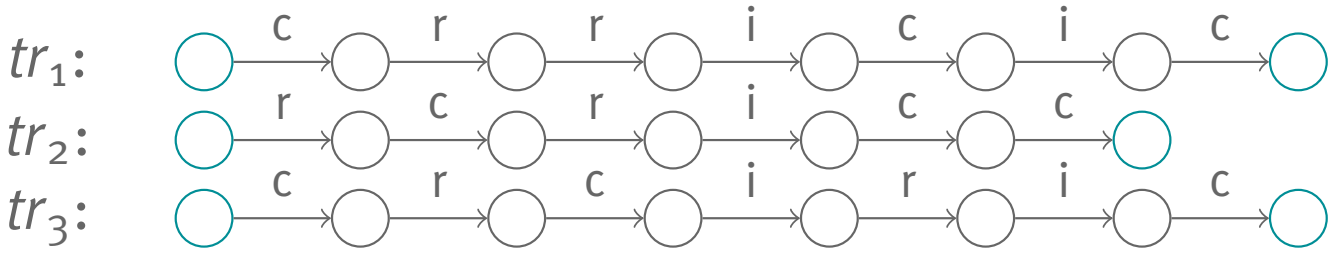
A reduction is also possible for formulae of interest:

$$\varphi_{GNI} = \forall \pi_1 . \forall \pi_2 . \exists \pi_3 . \mathcal{G}(\bigwedge_{a \in L} [a]_{\pi_1} \leftrightarrow [a]_{\pi_3}) \wedge \mathcal{G}(\bigwedge_{a \in H} [a]_{\pi_2} \leftrightarrow [a]_{\pi_3})$$

Pushdown Model Checking - Well-alignedness

Idea: align the stack actions and use visibly pushdown automata (VPA)

Full alignment on all positions is too restrictive; well-alignedness: synchronise on observation points, allow some divergence in between

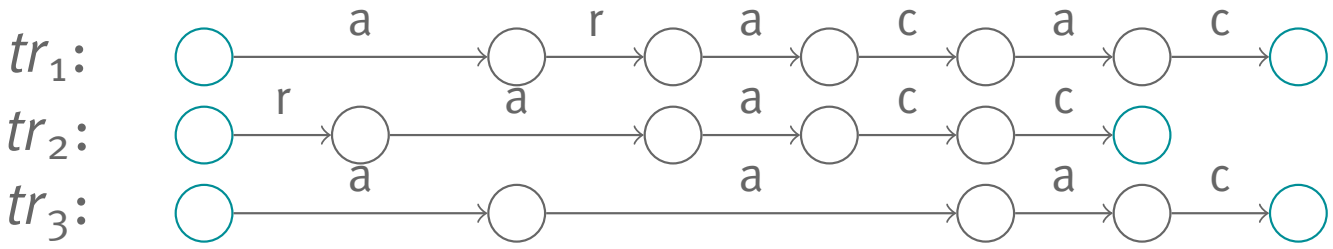


subtraces between blue positions on tr_1 and tr_2 are well-aligned ✓
subtraces between blue positions on tr_1 and tr_3 are not well-aligned ✗

Pushdown Model Checking - Well-alignedness

Idea: align the stack actions and use visibly pushdown automata (VPA)

Full alignment on all positions is too restrictive; well-alignedness: synchronise on observation points, allow some divergence in between

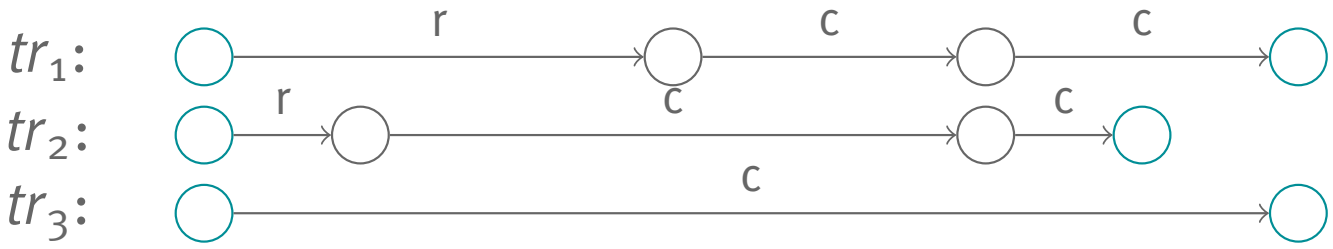


subtraces between blue positions on tr_1 and tr_2 are well-aligned ✓
subtraces between blue positions on tr_1 and tr_3 are not well-aligned ✗

Pushdown Model Checking - Well-alignedness

Idea: align the stack actions and use visibly pushdown automata (VPA)

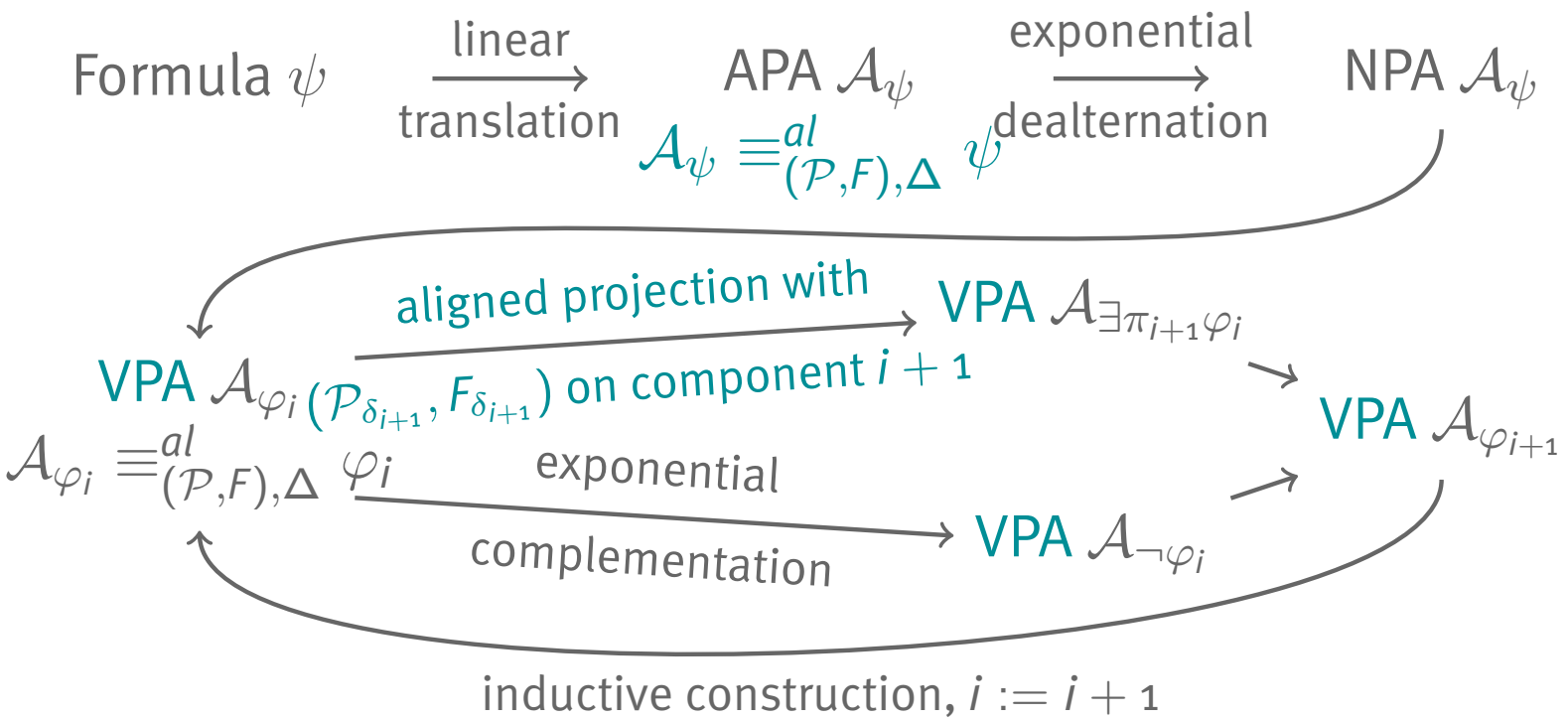
Full alignment on all positions is too restrictive; well-alignedness: synchronise on observation points, allow some divergence in between



subtraces between blue positions on tr_1 and tr_2 are well-aligned ✓
subtraces between blue positions on tr_1 and tr_3 are not well-aligned ✗

(Fair)Pushdown Model Checking Mumbling H_μ

Fair Pushdown Model Checking with basis *AP*:



$\mathcal{K} \models \varphi$ iff $\mathcal{L}(\mathcal{A}_\varphi) \neq \emptyset$; Model Checking in $(k+1)$EXPTIME

Conclusion

We explored different specification logics for the use in Model Checking:

- LTL and CaRet for single trace properties
- HyperLTL for synchronous hyperproperties
- H_μ for general asynchronous hyperproperties
- Stuttering H_μ for asynchronous hyperproperties with known stuttering criteria
- Mumbling H_μ for asynchronous hyperproperties with known observation criteria and pushdown models