Runtime Verification of Hyperproperties for Deterministic Programs

Srinivas Pinisetty, Dave Sands, **Gerardo Schneider**

Dept. of Computer Science and Engineering

Chalmers | University of Gothenburg

Sweden

UNIVERSITY OF GOTHENBURG

Vetenskapsrådet

FormaliSE@ICSE
Gothenburg, Sweden
2 June 2018

General Data Protection Regulation* (GDPR)

"Personal data must be: **adequate**, **relevant**, and limited to **the minimum necessary** in relation to the **purposes** for which they are processed…"

Simplifying: "You should not **collect** more data than what is strictly required for the **intended computation**"

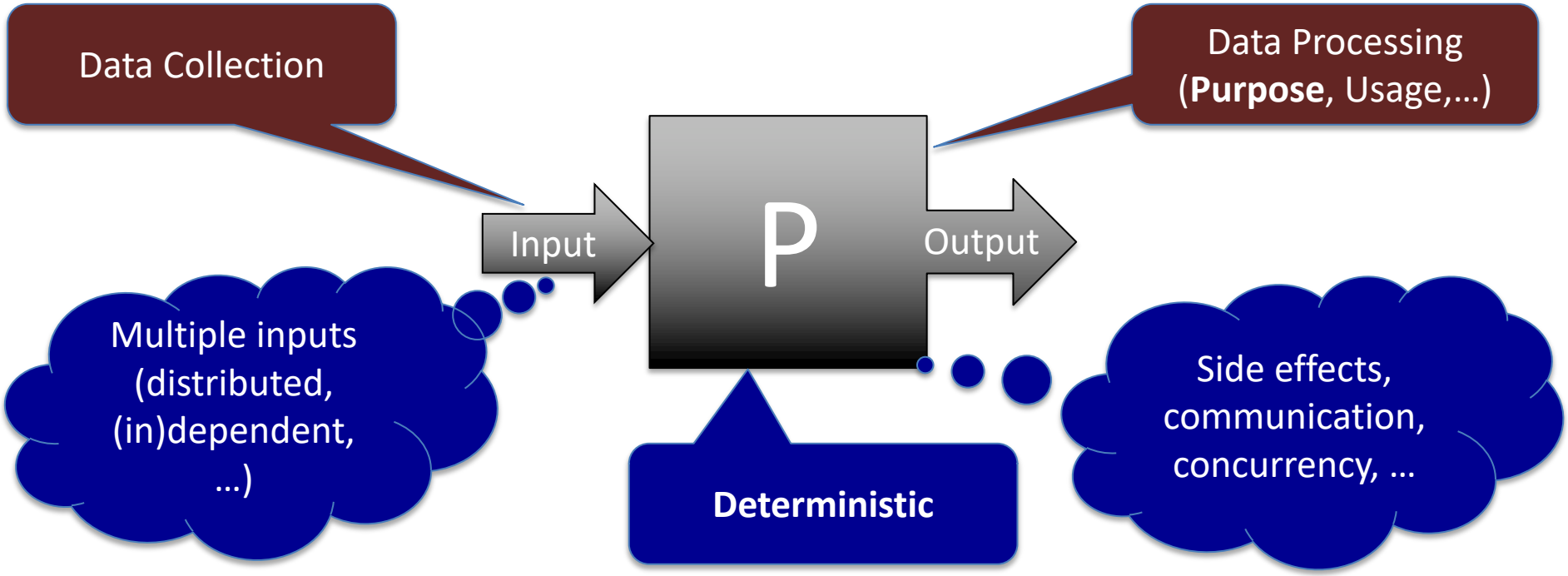
- What might this mean? How can it be ensured?
  - Statically? At runtime?

* EU – 2016/679: Entered into application 25 May 2018

What is the connection with the title of our paper?

- The above ”principle” is called *Data Minimization*
- Data minimization is a representative of an interesting class of properties
- Hyperproperties are properties defined over *sets of sets* of traces
  - ”Normal” properties are defined over *sets* of traces

Data Minimization

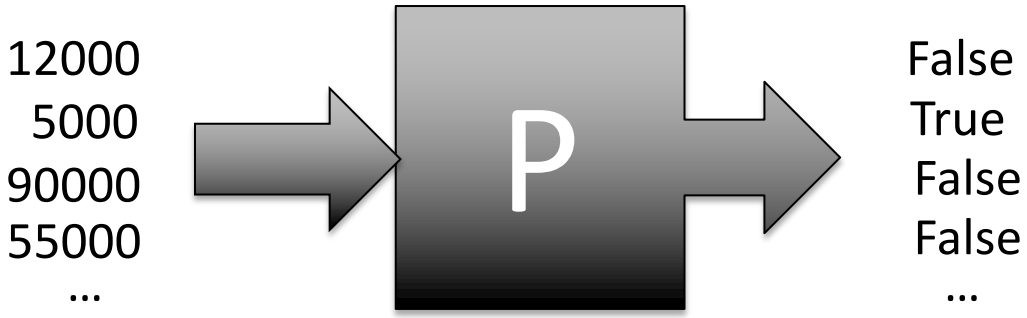


Collection vs Usage: We focus on the first (restricted)

- **Previous work**: Definitions + what can be done statically
  - T. Antignac, D. Sands, and G. Schneider. *Data Minimisation: A Language-Based Approach.* In SEC'17, vol 502 of IFIP AICT, pp. 442-456, 2017
- **Goal of this work**: Monitoring data minimization and other similar hyperproperties

A Simple Program…

```
1 input(salary);
2 benefits := (salary < 10000);
3 output(benefits);
```



Is the information about the salary really needed?

Data Minimization (Minimality)

Definition

P is data minimal if its output is ***totally dependent*** on its inputs: **any** variation of input *x* causes variation in output *y*

Two variants:

Monolithic: a single input source

Distributed: multiple independent sources

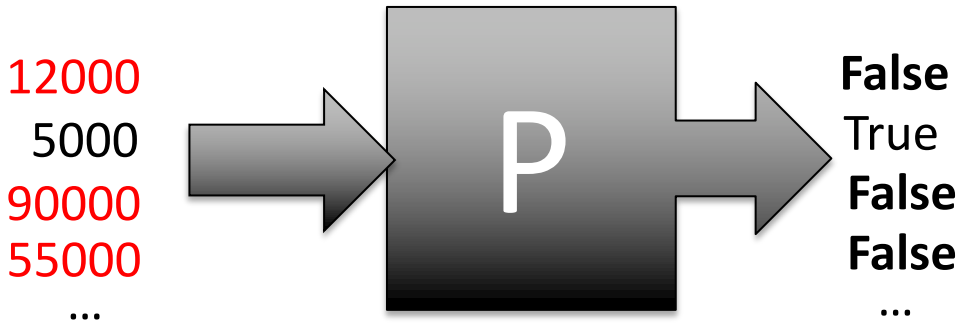
Monolithic case: minimality is just *injectivity*

Data Minimization (Minimality)

Definition

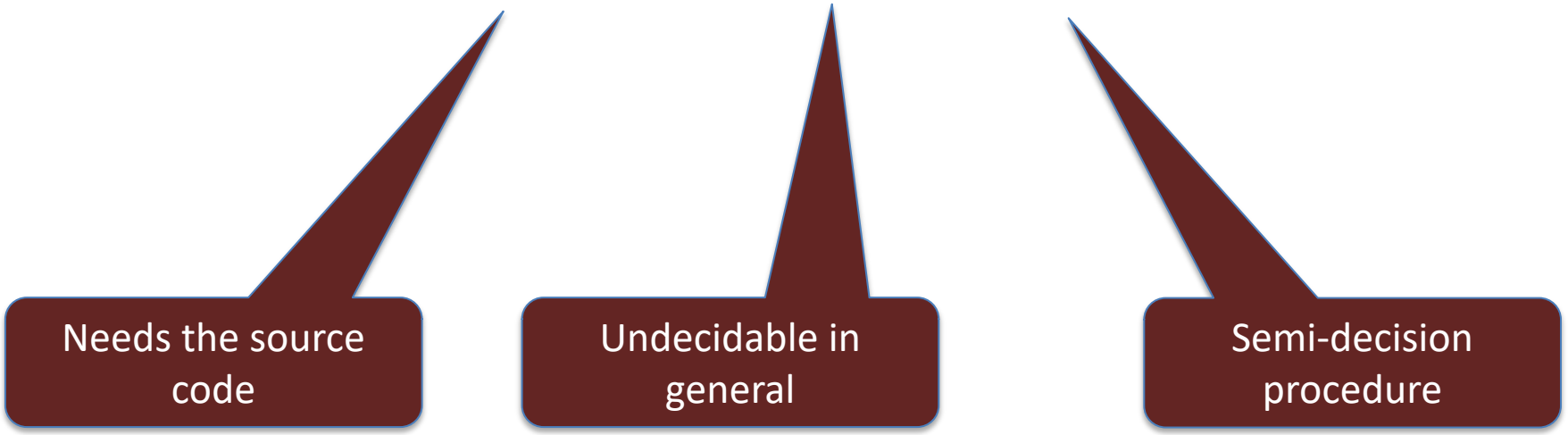
P is data minimal if its output is ***totally dependent*** on its inputs: **any** variation of input *x* causes variation in output *y*

P (the "benefits" program) is not (monolithical) minimal



Why Monitoring?

Statically detecting and ensuring (monolithic and distributed) data minimality is not easy*



* T. Antignac, D. Sands, and G. Schneider. *Data Minimisation: A Language-Based Approach.* In SEC'17, vol 502 of IFIP AICT, pp. 442-456, 2017

Can we do it?

- M.R. Clarkson, B. Finkbeiner, M. Koleini, K.K. Micinski, M.N. Rabe, C. Sánchez: *Temporal Logics for Hyperproperties*. POST'14
- B. Bonakdarpour and B. Finkbeiner. 2016. *Runtime Verification for HyperLTL.* In RV'16.
- S. Agrawal and B. Bonakdarpour. 2016. *Runtime Verification of k-Safety Hyperproperties in HyperLTL.* In CSF'16.
- N. Brett, U. Siddique, and B. Bonakdarpour. 2017. *Rewriting-Based Runtime Verification for Alternation-Free HyperLTL.* In TACAS'17.

Monitoring algorithms for the **alternation-free** fragment of **HyperLTL**

Universal quantifiers: usually **not** monitorable

Existential quantifiers: monitorable

What Does it Means for Us?

Data minimization may be expressed in HyperLTL!

We are done then! Somebody else did it!

Yes, but… Algorithms are general for HyperLTL

(Monolithic) Data Minimization Revisited…

Quantification over traces

$$\forall \pi, \forall \pi' : \pi_I \neq \pi'_I \implies \pi_O \neq \pi'_O.$$

HyperLTL$_{2S}$

$$\forall \pi, \forall \pi' : \psi$$
$$\psi ::= a_\pi \mid \neg\psi \mid \psi \vee \psi$$

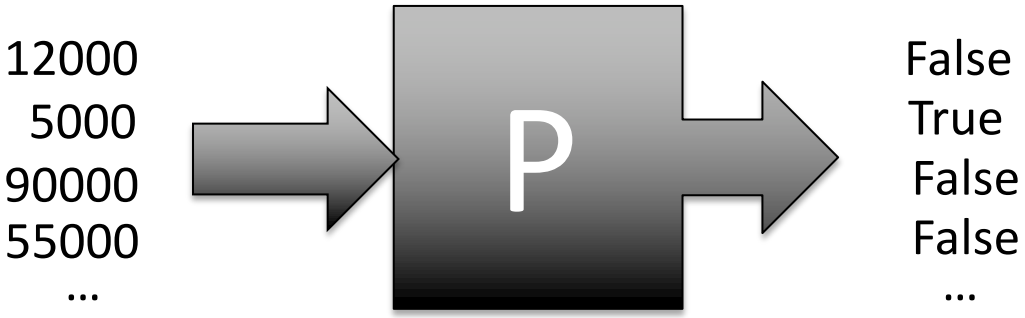
- ***Non-minimality*** *is* monitorable but it is in general impossible to give a final verdict for minimality!
- Traces are of fixed length (one)
- We are considering deterministic programs
- The property only talks about inputs and outputs!

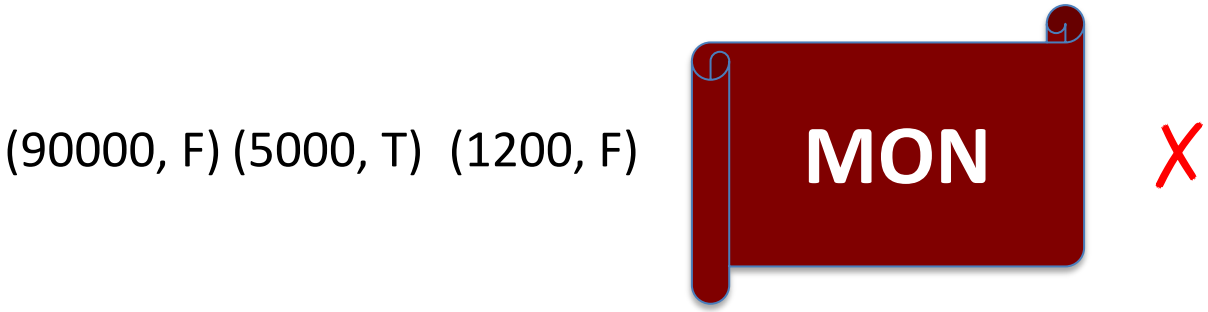
It should be simpler to monitor!

Monitoring Data Minimization (Program-in-loop)

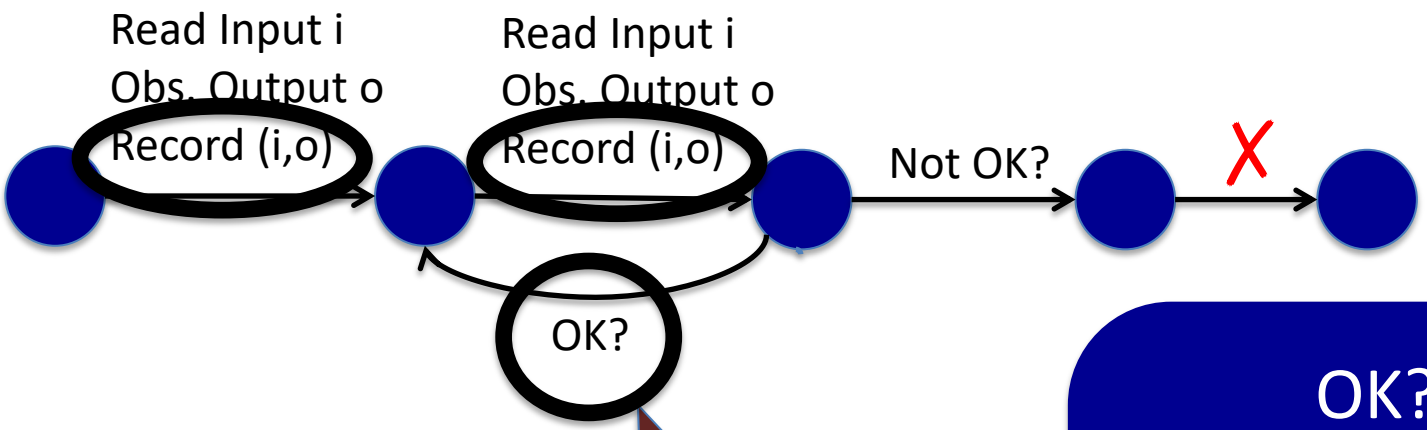
Reduction to a *trace property*

Not very important (algorithm is simpler)





Monitor for Data Minimization



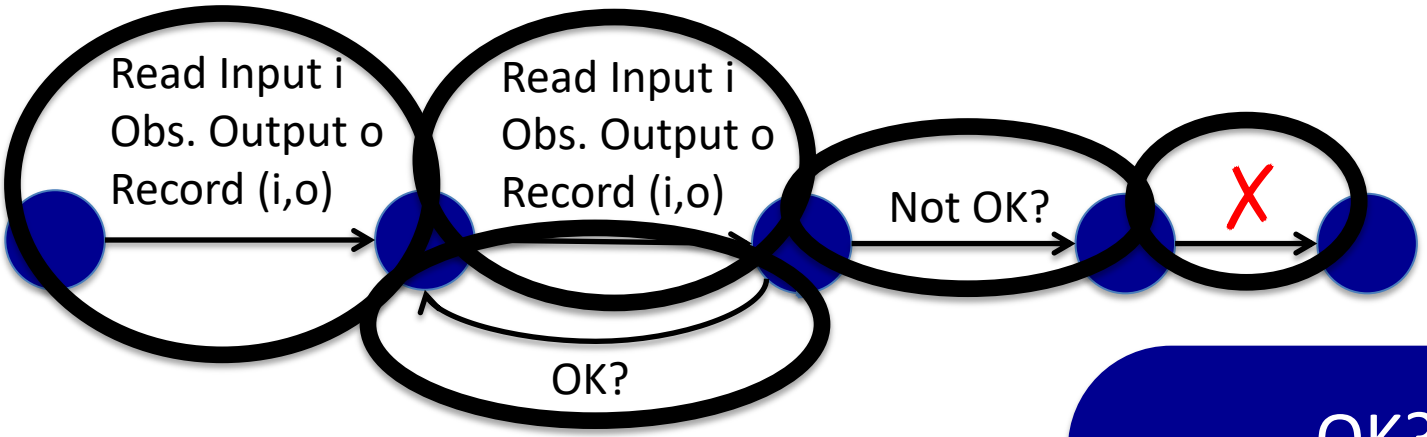
Input	Output
i_1	o_1
i_2	o_2
i_3	o_3
…	…

The "intelligence" is in the **OK?** predicate

OK?
(**Monolithic** Data Minimization)

Is there a prefix with the same output and different input?

Monitor for ***Monolithic*** Data Minimization



Input	Output
1200	F
5000	T
90000	T

OK?
(**Monolithic** Data Minimization)

Is there a prefix with the same output and different input?

Monitor for ***Strong Distributed*** Data Minimization

- Monitor very similar to monolithic case but reading inputs from all sources independently
  - More states to read from all sources
- The **OK?** predicate will be different

- (A bit more complex – more theoretical results in Tech Rep *Monitoring Data Minimisation*)

Is that All?

Other (hyper)properties *similar* to Data Minimization

- *Non-interference*
- *Integrity*
- Software doping (or rather *doping-free programs*)

Non-interference

- *P* satisfies non-interference if every pair of traces with the same (initial) *low* observation remains indistinguishable for low users

- Absence of strong dependency between input *secret (high)* and output *public (low)*
  - The (public) output observed by the low security users should only depend on low input information

$$\forall \pi, \forall \pi' : (\pi_{I,L} = \pi'_{I,L}) \implies (\pi_{O,L} = \pi'_{O,L})$$

Integrity

- Integrity requires that *high* behaviour of a system should not be influenced by *low* inputs (that can be potentially altered by a malicious user)

- Traces having the same high inputs but possibly different low inputs should have the same high outputs

$$\forall \pi, \forall \pi' : (\pi_{I,H} = \pi'_{I,H}) \implies (\pi_{O,H} = \pi'_{O,H})$$

Doping-Free Programs

- *P* is doping-free if small variations in the input produces small variations in the output

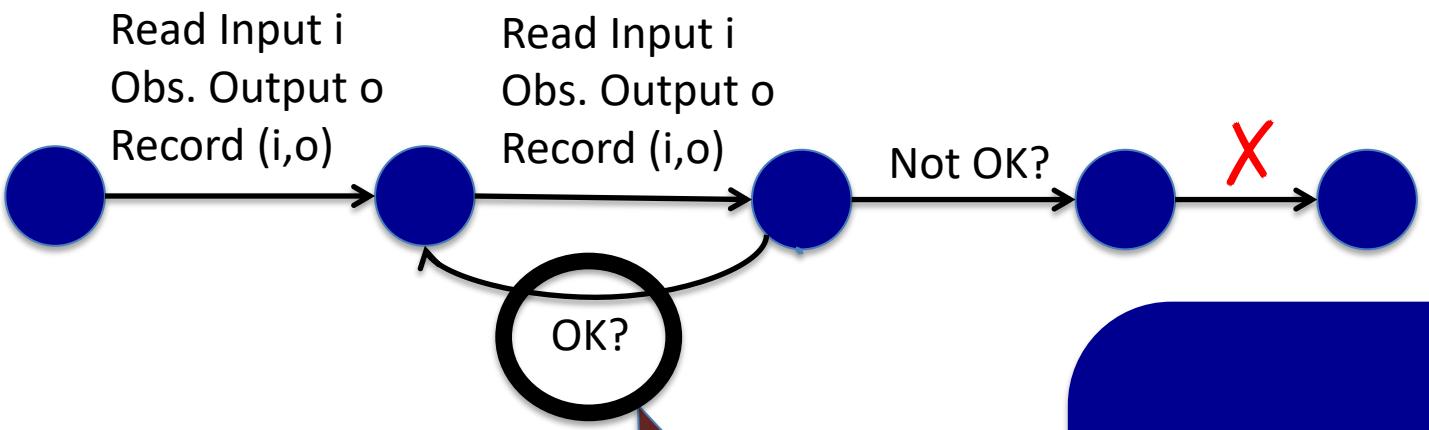
- A *parameterized* program *P* is doping-free if for all pairs of parameters of interest *p* and *p'*, and input *i*, then $P_p(i) = P_{p'}(i)$

$$\forall \pi, \forall \pi' : ((\pi_{Parm} \in PIntrs) \wedge (\pi'_{Parm} \in PIntrs) \wedge (\pi_I = \pi'_I)) \implies (\pi_O = \pi'_O)$$

Other (hyper)properties *similar* to Data Minimization

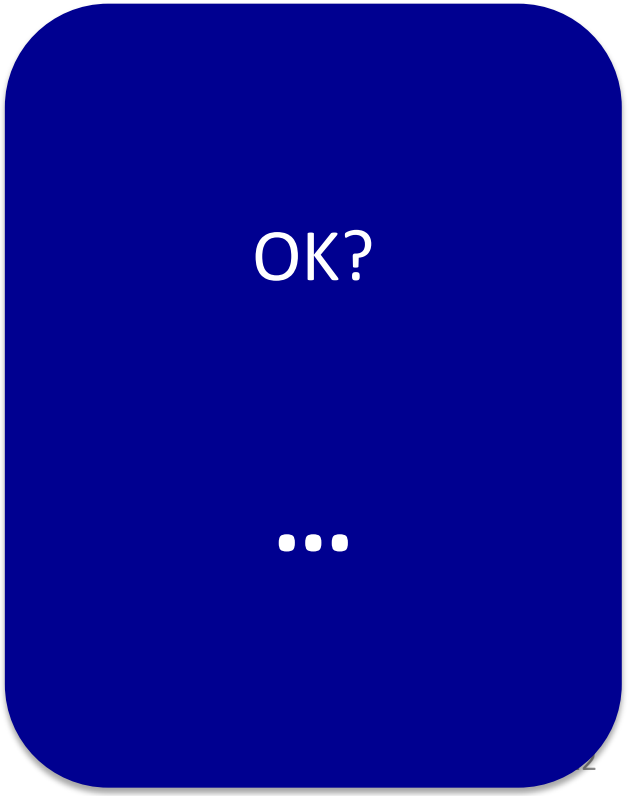
Property	Property expressed in Hyper_{2S}
Data minimisation (Monolithic minimality)	$\forall \pi, \forall \pi' : \pi_I \neq \pi'_I \implies \pi_O \neq \pi'_O.$
Non-Interference	$\forall \pi, \forall \pi' : (\pi_{I,L} = \pi'_{I,L}) \implies (\pi_{O,L} = \pi'_{O,L})$
Integrity	$\forall \pi, \forall \pi' : (\pi_{I,H} = \pi'_{I,H}) \implies (\pi_{O,H} = \pi'_{O,H})$
Software doping (doping free program)	$\forall \pi, \forall \pi' :$ $((\pi_{Parm} \in PIntrs) \wedge (\pi'_{Parm} \in PIntrs) \wedge (\pi_I = \pi'_I)) \implies (\pi_O = \pi'_O)$
Strong distributed minimality	$\forall \pi, \forall \pi',$ let $\pi_I = (i_1, \cdots, i_n), \pi'_I = (i'_1, \cdots, i'_n).$ $(\exists x \in [1,n] : i_x \neq i'_x \wedge$ $\forall y \in [1,n] : y \neq x \implies i_y = i'_y) \implies \pi_o \neq \pi'_o.$

Monitor for other Hyperproperties in $HyperLTL_{2S}$



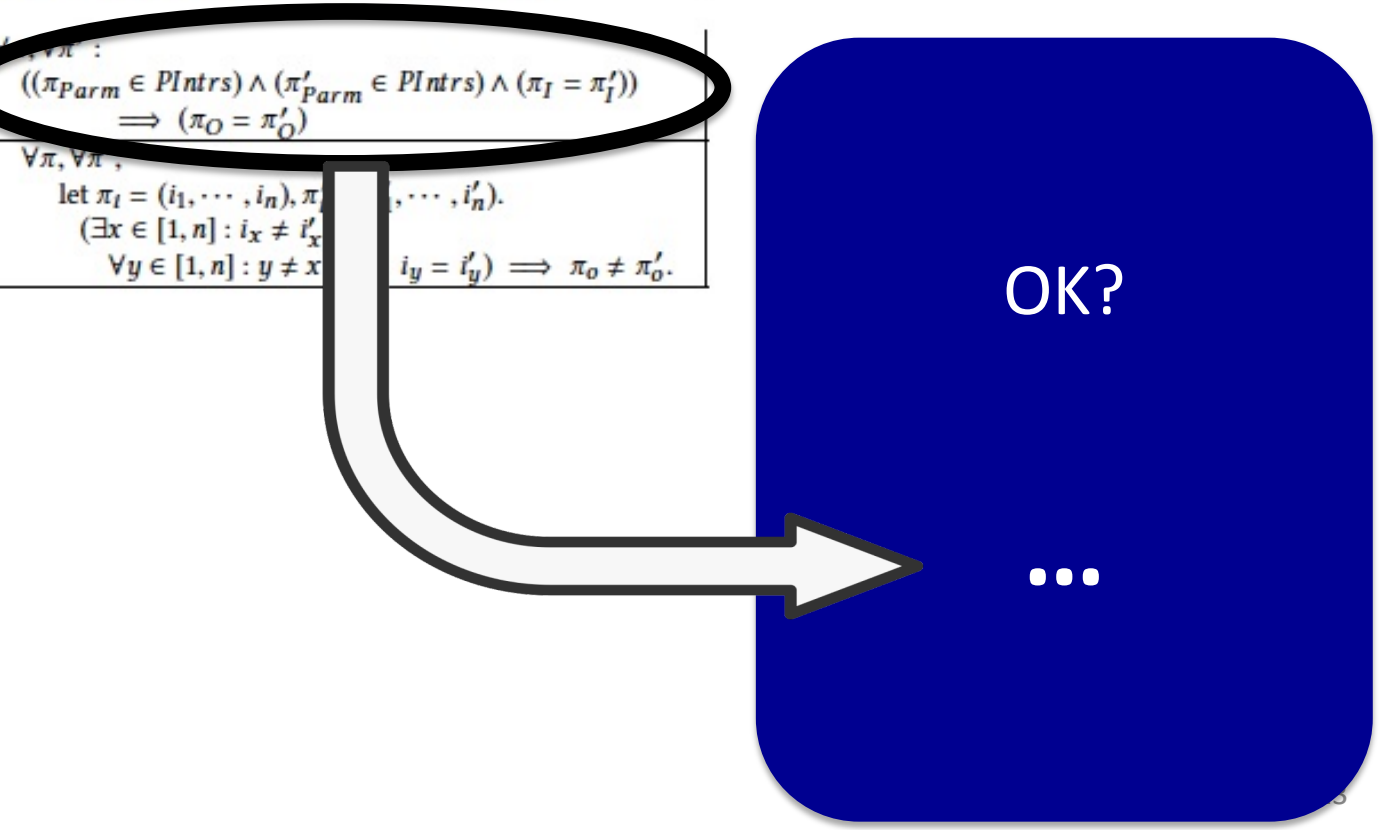
Input	Output
i_1	o_1
i_2	o_2
i_3	o_3
…	…

The "intelligence" is in the **OK?** predicate



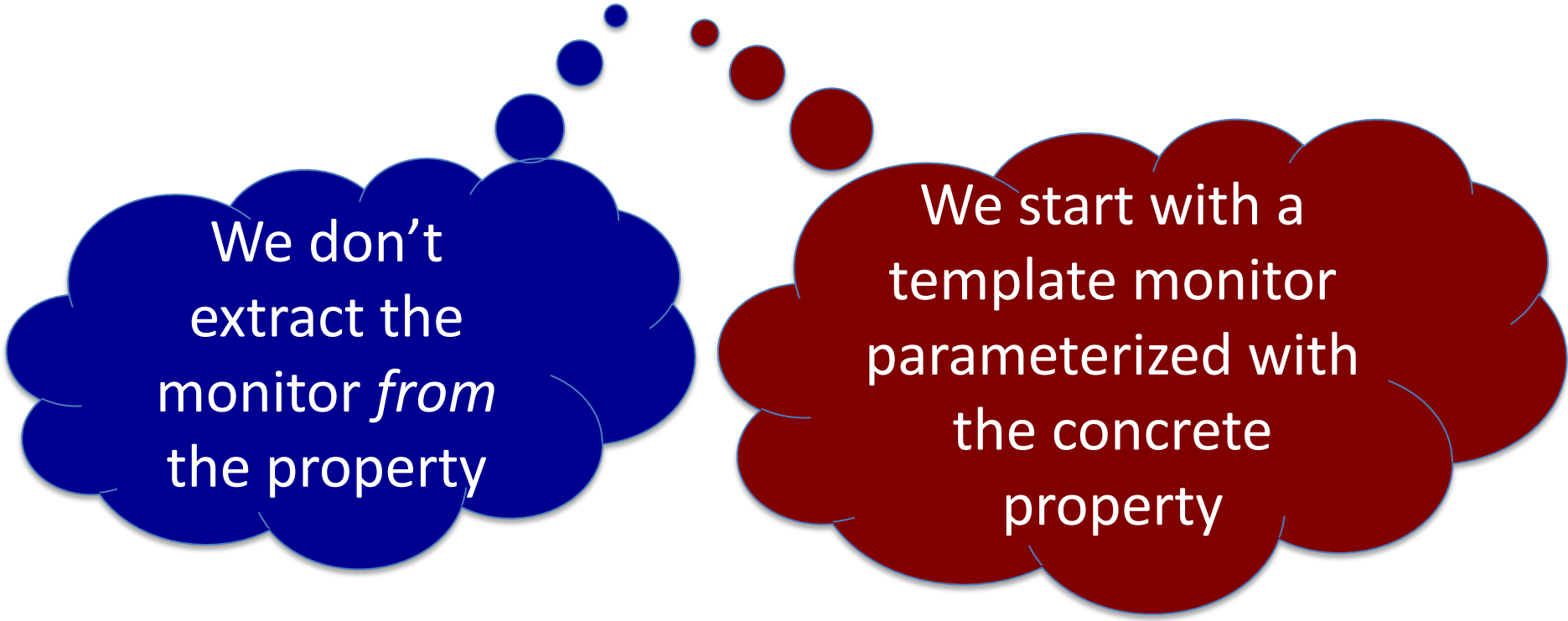
Parameterized Monitor for Hyperproperties in HyperLTL$_{2S}$

Property	Property expressed in Hyper$_{2S}$
Data minimisation (Monolithic minimality)	$\forall\pi,\forall\pi' : \pi_I \neq \pi'_I \implies \pi_O \neq \pi'_O.$
Non-Interference	$\forall\pi,\forall\pi' : (\pi_{I,L} = \pi'_{I,L}) \implies (\pi_{O,L} = \pi'_{O,L})$
Integrity	$\forall\pi,\forall\pi' : (\pi_{I,H} = \pi'_{I,H}) \implies (\pi_{O,H} = \pi'_{O,H})$
Software doping (doping free program)	$\forall\pi,\forall\pi' :$ $((\pi_{Parm} \in PIntrs) \wedge (\pi'_{Parm} \in PIntrs) \wedge (\pi_I = \pi'_I)) \implies (\pi_O = \pi'_O)$
Strong distributed minimality	$\forall\pi,\forall\pi'$, let $\pi_I = (i_1,\cdots,i_n), \pi$ $,\cdots,i'_n)$. $(\exists x \in [1,n] : i_x \neq i'_x$ $\forall y \in [1,n] : y \neq x$ $i_y = i'_y) \implies \pi_o \neq \pi'_o.$

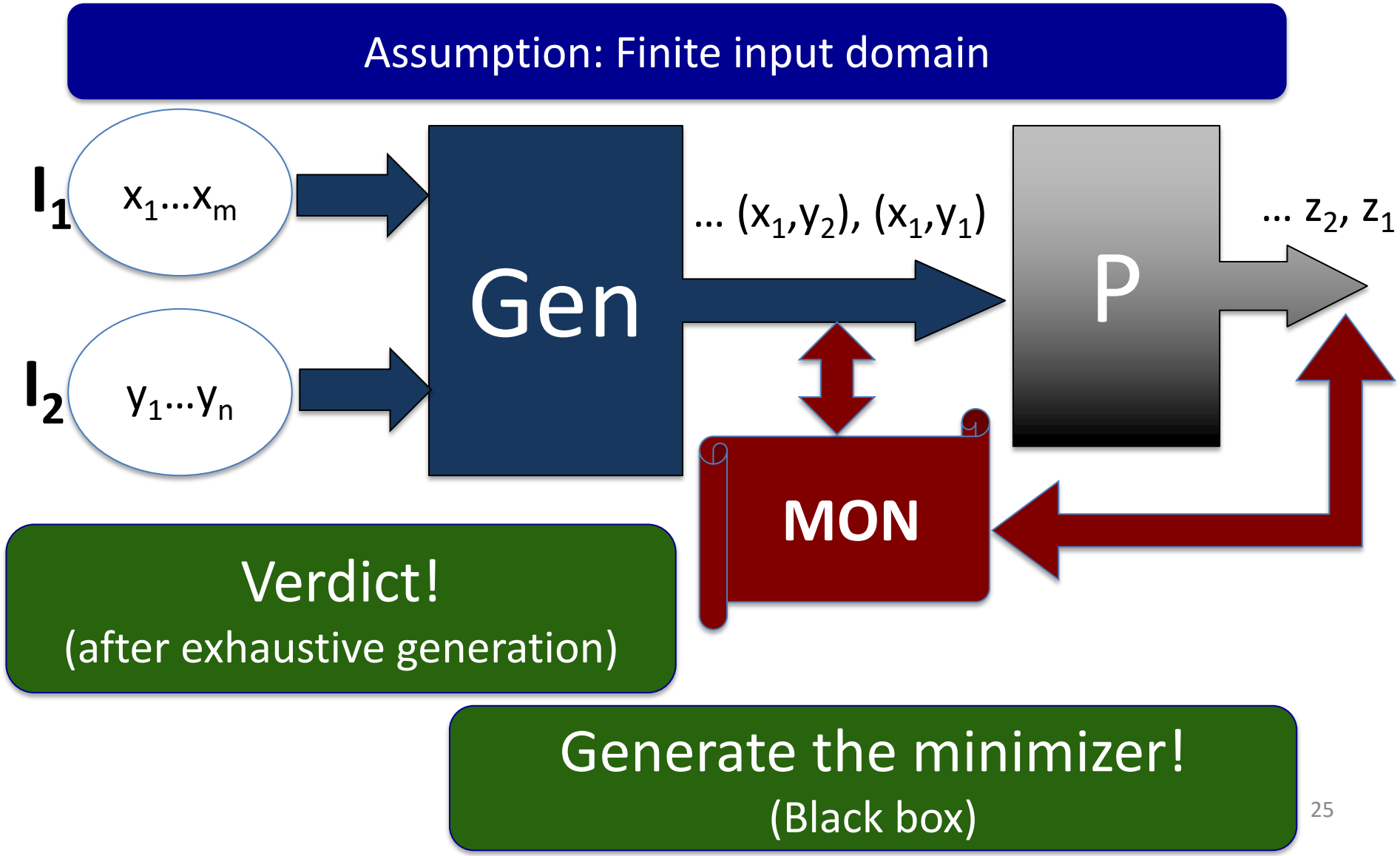


Runtime Verification (Monitoring) but…
(an unimportant clarifying note)

- The RV technique we are using here doesn't follow the "standard" way of getting the monitor



(Controlled) Offline Monitoring Data Minimization



Summary

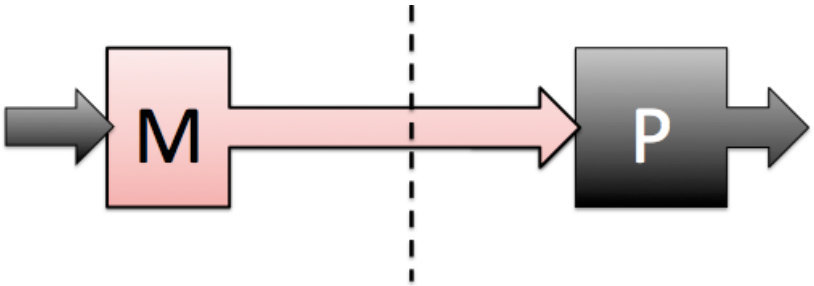
- Parameterized monitor for **HyperLTL$_{2S}$**
- Online monitorability for violations of property
  - Generalizes to traces of fixed length (not only 1)
  - Order of the traces not important (they may be reordered)
- Complexity: quadratic in the length of the observed trace
- Offline monitoring under assumption of finite input domains
  - Decidable (trivial!)
  - For data minimization: extraction of a minimizer
  - Optimizations are possible (taking into account size of output domain, etc)

Questions?

M is a *monolithic pre-processor* for P iff

$P \circ M = P$

$M \circ M = M$



Distributed Minimiser