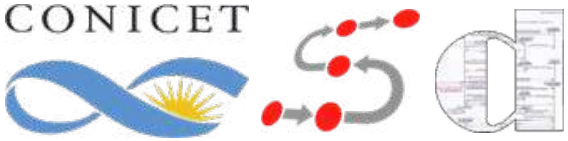
Is your software on dope?

Formal analysis of surreptitiously "enhanced" programs

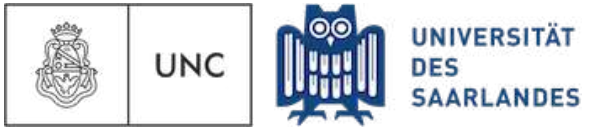
Gilles Barthe, Sebastian Biewer, Pedro R. D'Argenio,
Bernd Finkbeiner, and Holger Hermanns

IMDEA Software (ES) UN Córdoba – CONICET (AR) Saarland University (DE)

http://www.cs.famaf.unc.edu.ar/~dargenio/



FMT, UT, January 2017



Motivation

You get a third party technically compatible cartridge but …

- Refuses to work
- Shows a warning sign
- Informs “low toner” earlier than needed

Motivation

Refuses third party battery and chargers

Refuses or changes your vote!!!

Motivation

- "Chip tuning":

  The electronic control unit (ECU) is reprogrammed to change characteristics (e.g. power, emissions, fuel consumption)

Motivation

Volkswagen emissions scandal

A general characterization

- Why characterizations?

A general chara

Clearly not in the interest of the manufacturer

- Why characterizations?

To formulate and enforce rigid requirements on software driven by public interest, so as to effectively ban software doping.

A general chara

Clearly not in the interest of the manufacturer

- Why characterizations?

  To formulate and enforce rigid requirements on software driven by public interest, so as to effectively ban software doping.

- A software system is doped if:

  the manufacturer has included a hidden functionality in such a way that the resulting behaviour intentionally favors a designated party, against the interest of society or of the software licensee.

A general chara

Clearly not in the interest of the manufacturer

- Why characterizations?

  To formulate and enforce rigid requirements on software driven by public interest, so as to effectively ban software doping.

- A software system is doped if:

  the manufacturer has included a hidden functionality in such a way that the resulting behaviour intentionally favors a designated party, against the interest of society or of the software licensee.

Not possible to formalize

E.g. iPhone 6

CONICET

Doping by discrimination

```
procedure PRINTER(cartridge_info)

    READ(document)
    while PAGESTOPRINT(document) > 0 do
        READ(paper_available?)
        if ¬paper_available? then
            TURNON(alert_signal)
            WAITUNTIL(paper_available?)
            TURNOFF(alert_signal)
        end if
        PRINTNEXTPAGE(page_out,document)
    end while

end procedure
```

Doping by discrimination

```
procedure PRINTER(cartridge_info)

    READ(document)
    while PAGESTOPRINT(document) > 0 do
        READ(paper_available?)
        if ¬paper_available? then
            TURNON(alert_signal)
            WAITUNTIL(paper_available?)
            TURNOFF(alert_signal)
        end if
        PRINTNEXTPAGE(page_out,document)
    end while

end procedure
```

parameters

inputs

outputs

Doping by discrimination

A clean program

```
procedure PRINTER(cartridge_info)

    READ(document)
    while PAGESTOPRINT(document) > 0 do
        READ(paper_available?)
        if ¬paper_available? then
            TURNON(alert_signal)
            WAITUNTIL(paper_available?)
            TURNOFF(alert_signal)
        end if
        PRINTNEXTPAGE(page_out,document)
    end while

end procedure
```

parameters

inputs

outputs

Doping by discrimination

```
procedure PRINTER(cartridge_info)
    if BRAND(cartridge_info) = my-brand then
        READ(document)
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out,document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

parameters

inputs

outputs

Doping by discrimination

```
procedure PRINTER(cartridge_info)
    if BRAND(cartridge_info) = my-brand then
        READ(document)
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out,document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

parameters

inputs

outputs

A doped program

CONICET

UNIVERSITÄT DES SAARLANDES

Doping by discrimination

- A program is clean (or doping-free) if for every parameter of interest it exhibits the same visible outputs when supplied with the same inputs.

Doping by discrimination

- A program is clean (or doping-free) if for every parameter of interest it exhibits the same visible outputs when supplied with the same inputs.
- Formally:

$$S : \mathsf{Param} \to \mathsf{In} \to 2^{\mathsf{Out}}$$

non-deterministic

S is *clean* (or *doping-free*) if for all

$\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i} \in \mathsf{In}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Doping by discrimination

- A program is clean (or doping-free) if for every parameter of interest it exhibits the same visible outputs when supplied with the same inputs.
- Formally:

$$S : \mathsf{Param} \to \mathsf{In} \to 2^{\mathsf{Out}}$$

non-deterministic

S is *clean* (or *doping-free*) if for all

$\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i} \in \mathsf{In}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Defined by a contract

E.g. all compatible cartridges

CONICET

UNC

UNIVERSITÄT DES SAARLANDES

Doping by discrimination

```
procedure PRINTER(cartridge_info)

    READ(document)
    while PAGESTOPRINT(document) > 0 do
        READ(paper_available?)
        if ¬paper_available? then
            TURNON(alert_signal)
            WAITUNTIL(paper_available?)
            TURNOFF(alert_signal)
        end if
        PRINTNEXTPAGE(page_out, document)
    end while

end procedure
```

```
procedure PRINTER(cartridge_info)
    if BRAND(cartridge_info) = my-brand then
        READ(document)
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out, document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

Doping and extended functionality

```
procedure PRINTER(cartridge_info)
    READ(document)
    if ¬NEWTYPE(document) ∨ SUPPORTSNEWTYPE(cartridge_info) then
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out,document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

Doping and extend ality

The cartridge is standard

```
procedure PRINTER(cartridge_info)
    READ(document)
    if ¬NEWTYPE(document) ∨ SUPPORTSNEWTYPE(cartridge_info) then
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out,document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

Doping and extend ality

The cartridge is standard

```
procedure PRINTER(cartridge_info)
    READ(document)
    if ¬NEWTYPE(document) ∨ SUPPORTSNEWTYPE(cartridge_info) then
        while PAGESTOPRINT(document) > 0 do
            READ(paper_available?)
            if ¬paper_available? then
                TURNON(alert_signal)
                WAITUNTIL(paper_available?)
                TURNOFF(alert_signal)
            end if
            PRINTNEXTPAGE(page_out,document)
        end while
    else
        TURNON(alert_signal)
    end if
end procedure
```

The input is not

Doping and extended functionality

- A program is clean if for every parameter of interest it exhibits the same visible outputs when supplied with any possible standard input.
- Formally

$$S \text{ is } \textit{clean} \text{ if for all } \mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$$
$$\text{and } \mathsf{i} \in \mathsf{In} \cap \mathsf{StdIn}, \; S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$$

Doping and extended functionality

- A program is clean if for every parameter of interest it exhibits the same visible outputs when supplied with any possible standard input.
- Formally

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$

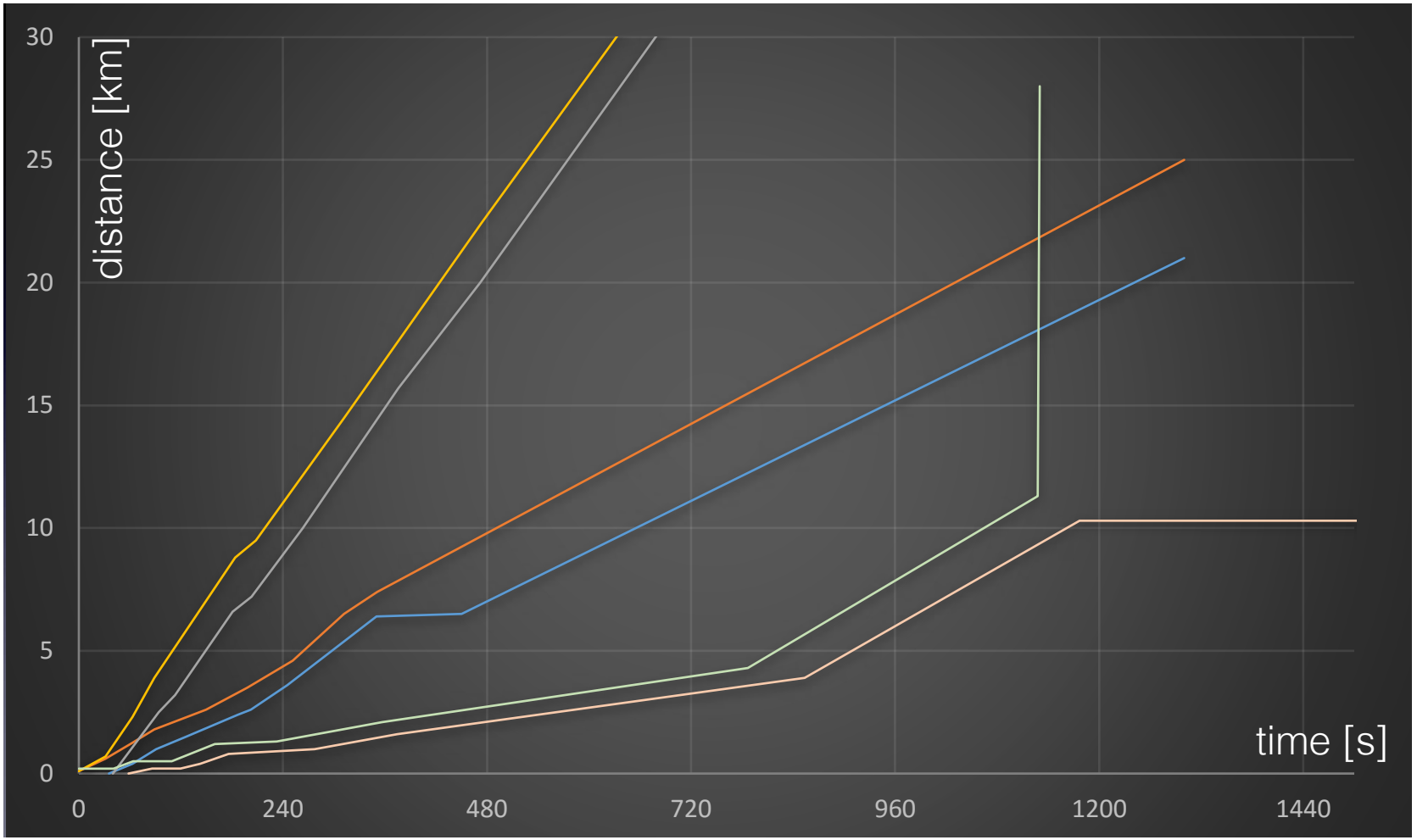
and $\mathsf{i} \in \mathsf{In} \cap \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Also defined by a contract

Doping by switching

- The Volkswagen emissions scandal:
  - The emission control part of the ECU regulates the emission of NOx (Mono-nitrogen oxides)
  - The selective catalytic reduction (SCR) model determines the diesel exhaust fluid (DEF) dosage
  - Volkswagen used two models
    - Standard
    - Alternate

Doping by switching



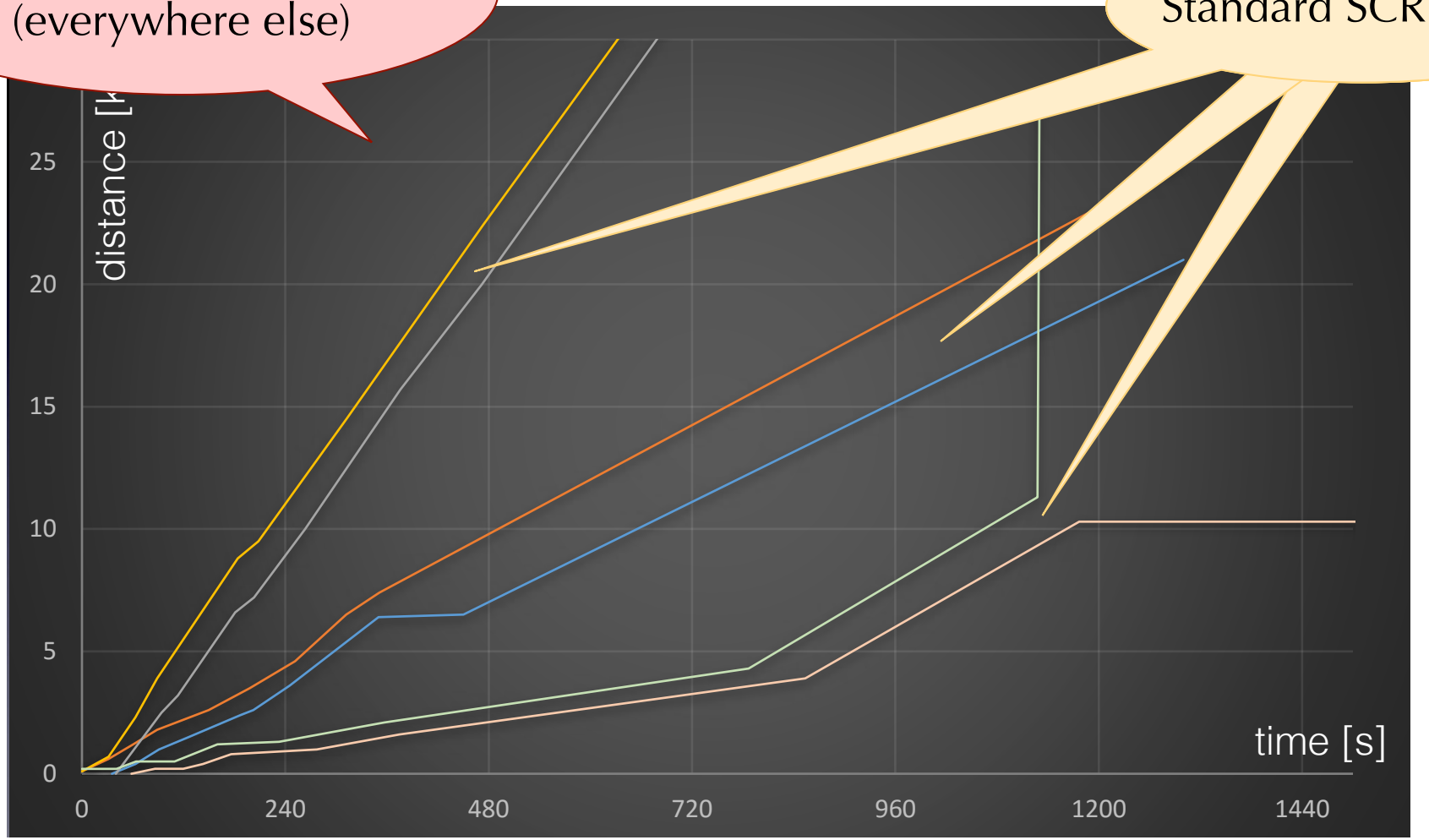
* Image from Felix Domke

The Volkswagen case

Doping by switching

Alternate SCR model
(everywhere else)

Standard SCR model



CONICET

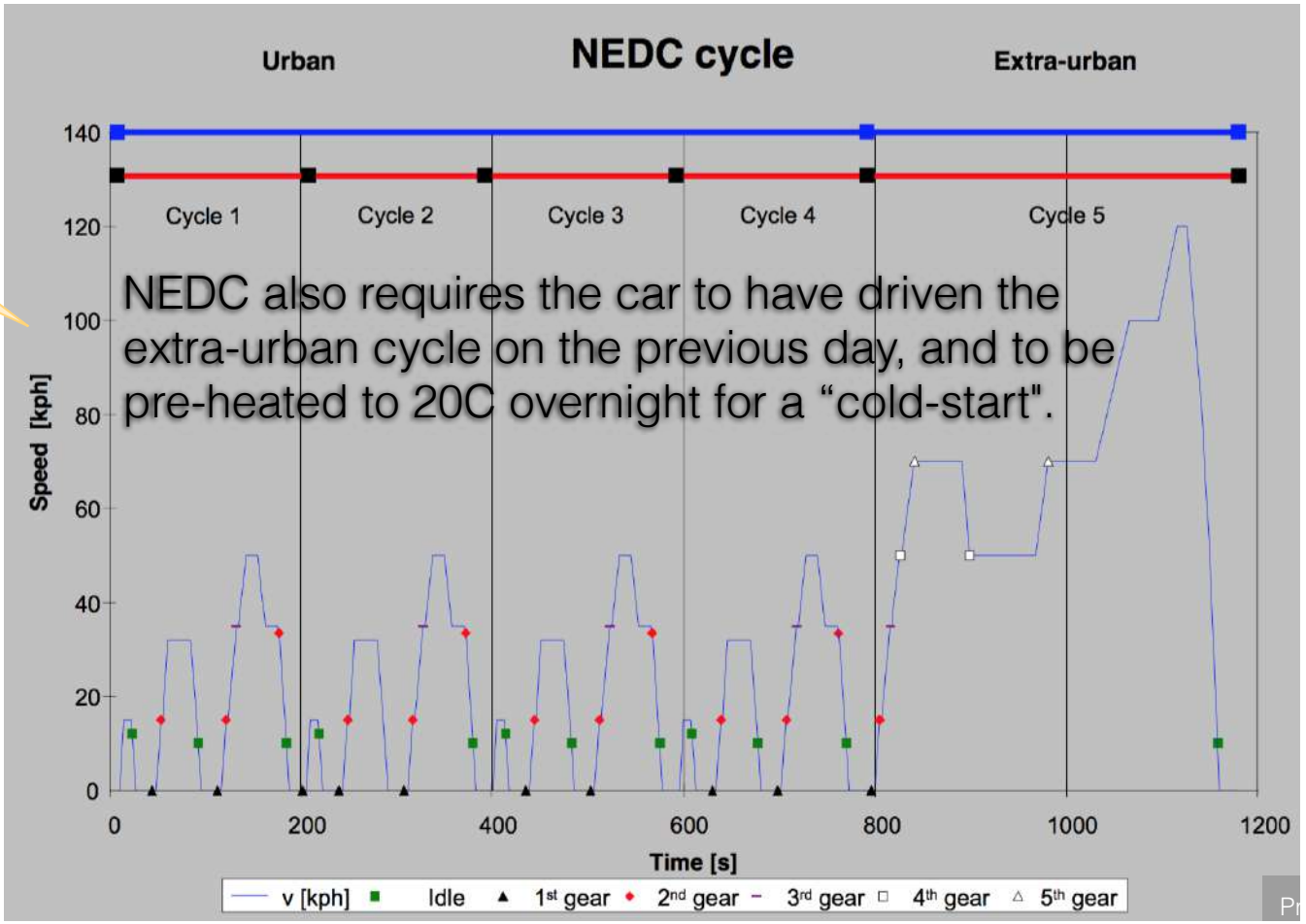
* Image from Felix Domke

UNC

UNIVERSITÄT DES SAARLANDES

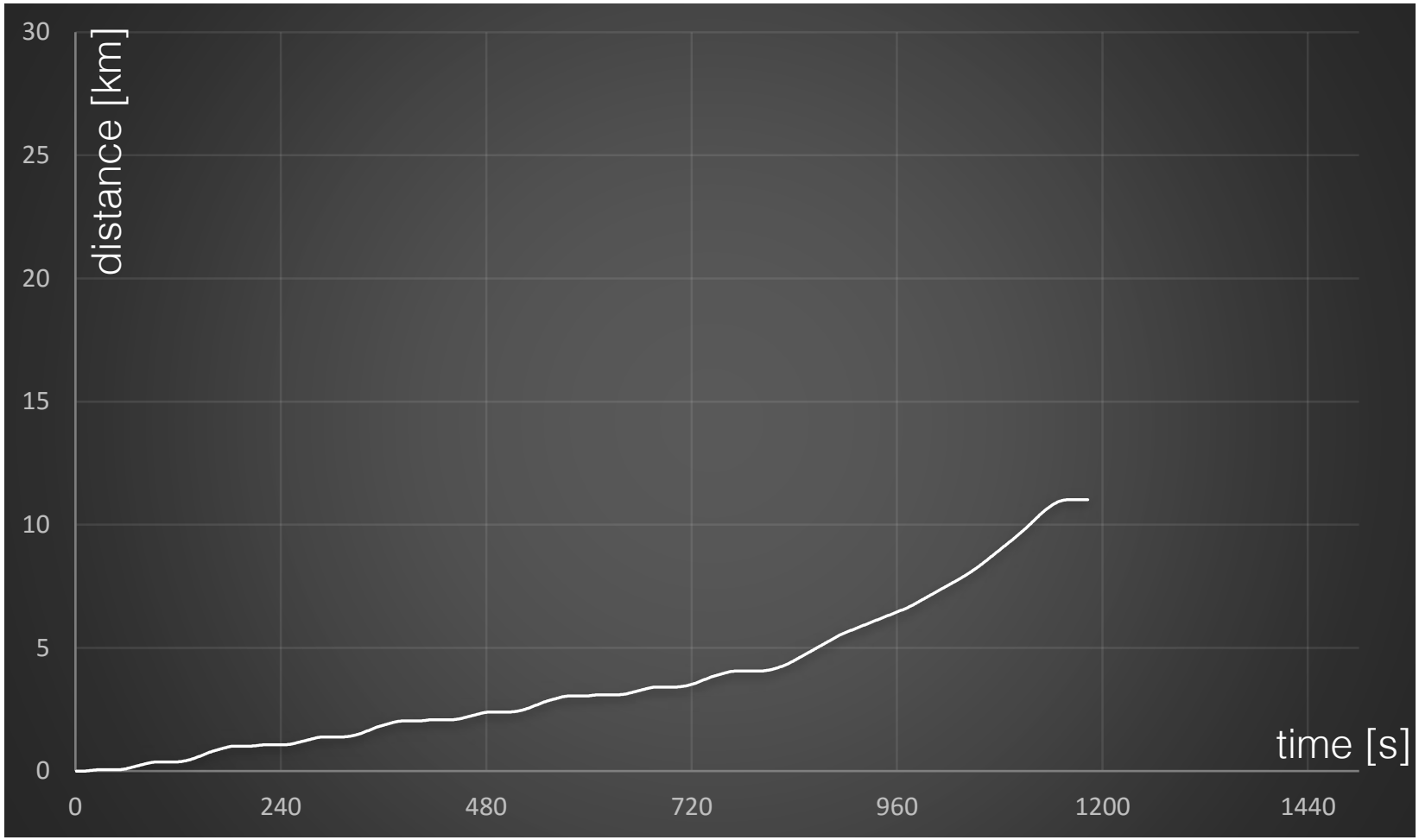
Doping by switching

Test for emission verification



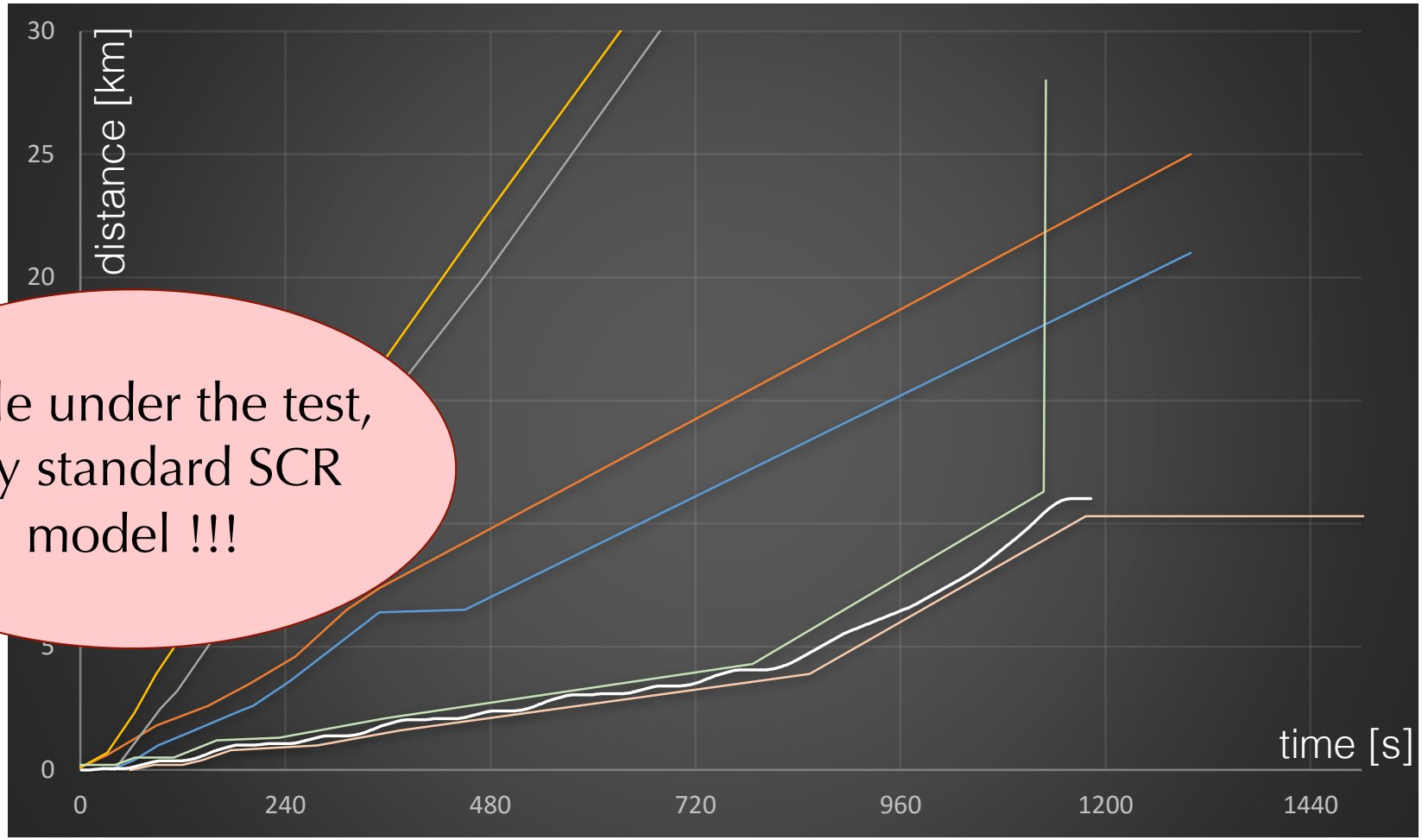
* Image from Felix Domke

Doping by switching



* Image from Felix Domke

Doping by switching



While under the test, only standard SCR model !!!

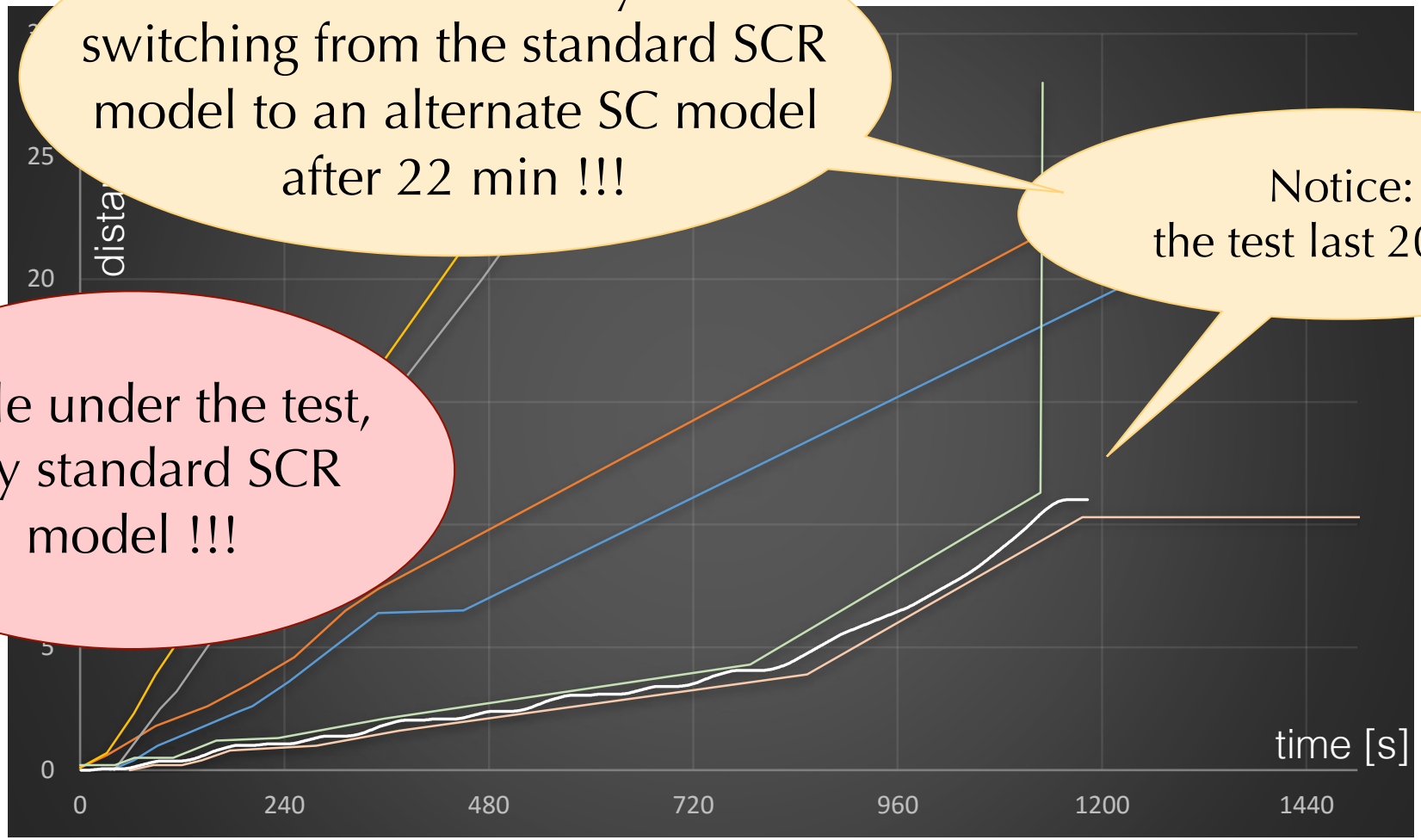
* Image from Felix Domke

Doping by switching

FIAT cheated by switching from the standard SCR model to an alternate SC model after 22 min !!!

Notice: the test last 20 min

While under the test, only standard SCR model !!!

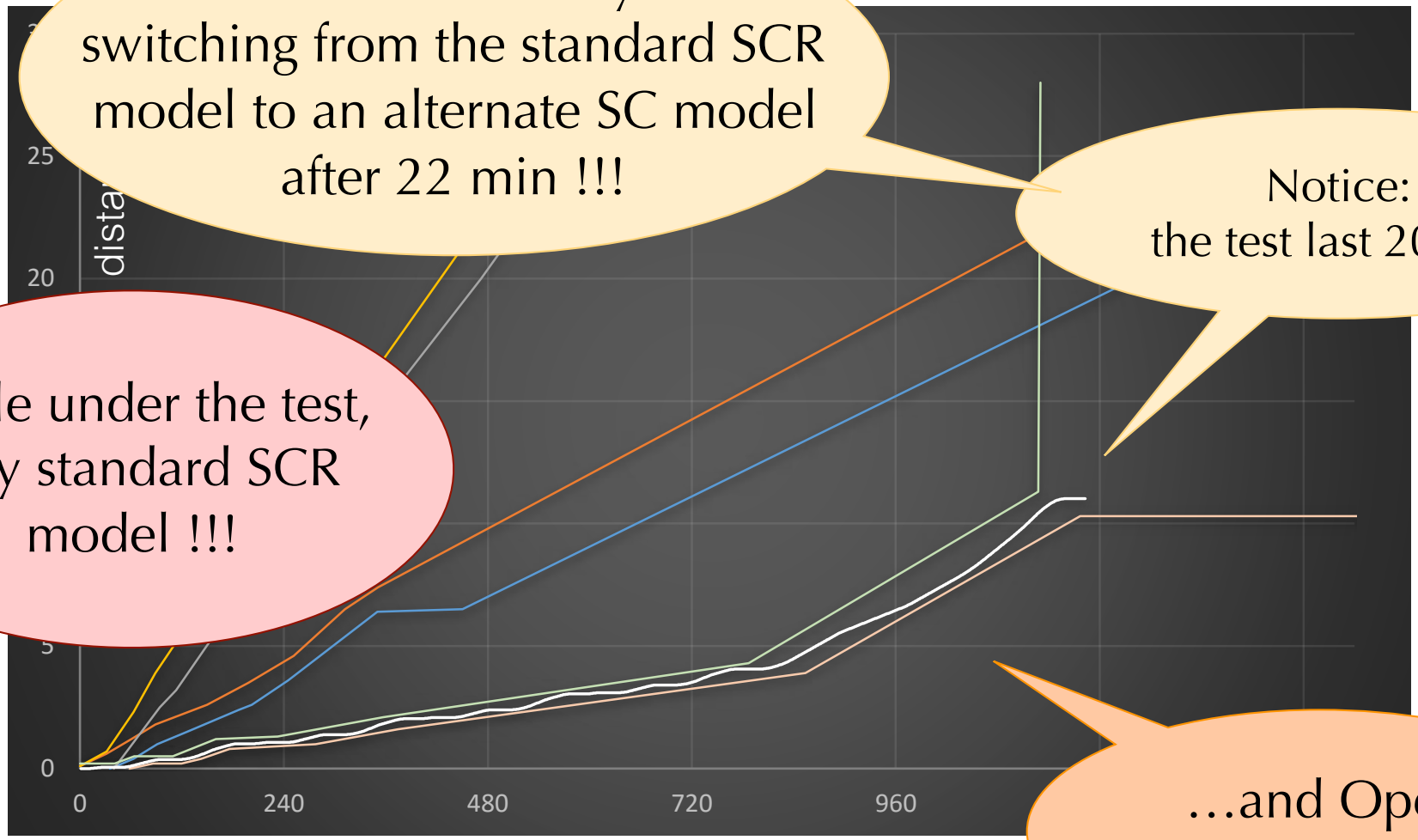


* Image from Felix Domke

Doping by switching

FIAT cheated by switching from the standard SCR model to an alternate SC model after 22 min !!!

Notice: the test last 20 min



While under the test, only standard SCR model !!!

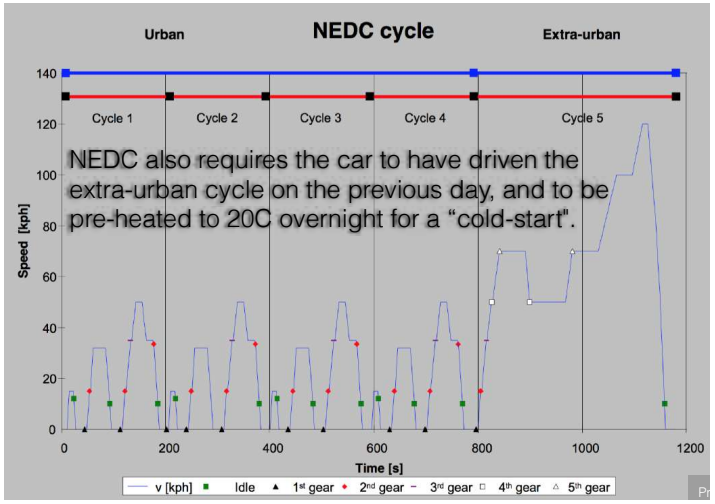
…and Opel also cheats!!

* Image from Felix Domke

Volkswagen and
FIAT and Opel and…

Doping by switching

- The standard inputs are defined by the test
- The spirit of the emission tests is to verify that the amount of NOx in the car exhaust gas does not go high in general
- Therefore one expects that:
  - if the input values deviates within a "reasonable distance" from the standard, the output values are also within a "reasonable distance" from the expected output value produced by the standard input



* Image from Felix Domke

Doping by switching

- Formally

S is *robustly clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$, whenever $\mathsf{i} \in \mathsf{StdIn}$ and $d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}') \leq \kappa_{\mathsf{i}}$,

1. for all $\mathsf{o} \in S(\mathsf{p})(\mathsf{i})$ there exists $\mathsf{o}' \in S(\mathsf{p}')(\mathsf{i}')$ such that $d_{\mathsf{Out}}(\mathsf{o}, \mathsf{o}') \leq \kappa_{\mathsf{o}}$, and
2. for all $\mathsf{o}' \in S(\mathsf{p}')(\mathsf{i}')$ there exists $\mathsf{o} \in S(\mathsf{p})(\mathsf{i})$ such that $d_{\mathsf{Out}}(\mathsf{o}, \mathsf{o}') \leq \kappa_{\mathsf{o}}$.

The distances and the thresholds are also defined by the contract

CONICET

UNC

UNIVERSITÄT DES SAARLANDES

Doping by switching

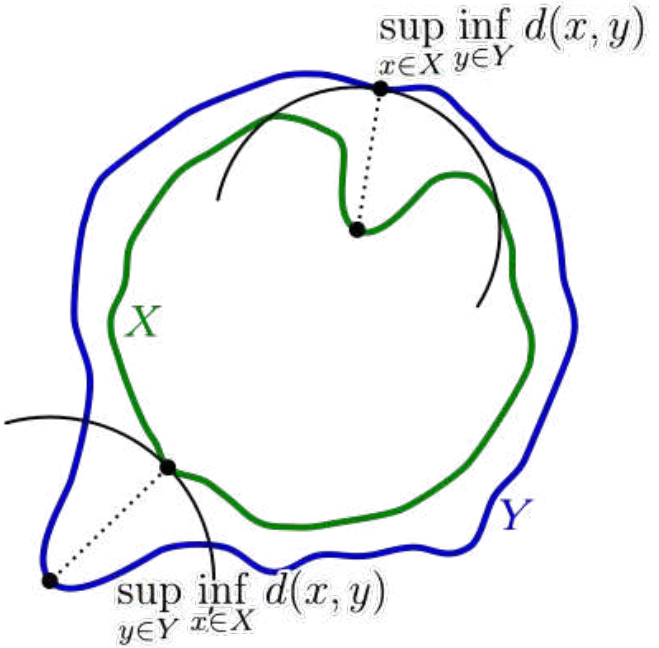
- Using Hausdorff distance

S is *robustly clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$, whenever $\mathsf{i} \in \mathsf{StdIn}$ and $d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}') \leq \kappa_{\mathsf{i}}$, then

$$\mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i}), S(\mathsf{p}')(\mathsf{i}')) \leq \kappa_{\mathsf{o}}$$

where

$$\mathcal{H}(d)(A, B) = \max \left\{ \begin{array}{l} \sup\limits_{a \in A} \inf\limits_{b \in B} d(a, b), \\ \sup\limits_{b \in B} \inf\limits_{a \in A} d(a, b) \end{array} \right\}$$



CONICET

Doping by switching

- Formally (more general)

S is *f-clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$, whenever $\mathsf{i} \in \mathsf{StdIn}$,

1. for all $\mathsf{o} \in S(\mathsf{p})(\mathsf{i})$ there exists $\mathsf{o}' \in S(\mathsf{p}')(\mathsf{i}')$ such that $d_{\mathsf{Out}}(\mathsf{o}, \mathsf{o}') \leq f(d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}'))$, and
2. for all $\mathsf{o}' \in S(\mathsf{p}')(\mathsf{i}')$ there exists $\mathsf{o} \in S(\mathsf{p})(\mathsf{i})$ such that $d_{\mathsf{Out}}(\mathsf{o}, \mathsf{o}') \leq f(d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}'))$.

Function f is also defined by the contract

Doping by switching

- Using Hausdorff distance

S is *f-clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$, whenever $\mathsf{i} \in \mathsf{StdIn}$,

$$\mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i}), S(\mathsf{p}')(\mathsf{i}')) \leq f(d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}'))$$

Analysis with Self-Composition

(deterministic programs)

$$\{\mathsf{PIntrst} \wedge \mathsf{StdIn} \wedge (\mathsf{PIntrst} \wedge \mathsf{StdIn})[\vec{x}/\vec{x}'] \wedge \vec{x}_{\mathsf{i}} = \vec{x}'_{\mathsf{i}}\}$$

$$S\,;S[\vec{x}/\vec{x}']$$

$$\{\ \vec{x}_{\mathsf{o}} = \vec{x}'_{\mathsf{o}}\ \}$$

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$
and $\mathsf{i} \in \mathsf{In} \cap \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Not quite right:
fails if S does not terminate
but $S[x/x']$ does

CONICET

UNC

Analysis with Self-Composition

(deterministic programs)

- S is *clean* iff

$$\left(\begin{array}{l} \quad (\mathsf{PIntrst} \wedge \mathsf{StdIn}) \\ \wedge\, (\mathsf{PIntrst} \wedge \mathsf{StdIn})[\vec{x}/\vec{x}'] \\ \wedge\, \vec{x}_{\mathsf{i}} = \vec{x}'_{\mathsf{i}} \\ \wedge\, \mathrm{wp}(S, \mathsf{true}) \end{array}\right) \Rightarrow\ \mathrm{wp}(S; S[\vec{x}/\vec{x}'], \vec{x}_{\mathsf{o}} = \vec{x}'_{\mathsf{o}})$$

Analysis with Self-Composition
(deterministic programs)

- S is *robustly clean* iff

$$\left(\begin{array}{l} \mathsf{PIntrst} \wedge \mathsf{StdIn} \\ \wedge\, \mathsf{PIntrst}[\vec{x}/\vec{x}'] \\ \wedge\, d_{\mathsf{i}}(\vec{x}_{\mathsf{i}}, \vec{x}'_{\mathsf{i}}) \leq \kappa_{\mathsf{i}} \\ \wedge\, \mathrm{wp}(S, \mathsf{true}) \end{array}\right) \Rightarrow \mathrm{wp}(S; S[\vec{x}/\vec{x}'], d_{\mathsf{Out}}(\vec{x}_{\mathsf{o}}, \vec{x}'_{\mathsf{o}}) \leq \kappa_{\mathsf{o}})$$

$$\left(\begin{array}{l} \mathsf{PIntrst} \wedge \mathsf{StdIn} \\ \wedge\, \mathsf{PIntrst}[\vec{x}/\vec{x}'] \\ \wedge\, d_{\mathsf{i}}(\vec{x}_{\mathsf{i}}, \vec{x}'_{\mathsf{i}}) \leq \kappa_{\mathsf{i}} \\ \wedge\, \mathrm{wp}(S[\vec{x}/\vec{x}'], \mathsf{true}) \end{array}\right) \Rightarrow \mathrm{wp}(S[\vec{x}/\vec{x}']; S, d_{\mathsf{Out}}(\vec{x}_{\mathsf{o}}, \vec{x}'_{\mathsf{o}}) \leq \kappa_{\mathsf{o}})$$

CONICET UNC UNIVERSITÄT DES SAARLANDES

Analysis with Self-Composition

(deterministic programs)

- S is *f-clean* iff for all Y

$$\left(\begin{array}{l} \mathsf{PIntrst} \wedge \mathsf{StdIn} \\ \wedge\, \mathsf{PIntrst}[\vec{x}/\vec{x}'] \\ \wedge\, f(d_{\mathsf{i}}(\vec{x}_{\mathsf{i}}, \vec{x}'_{\mathsf{i}})) = Y \\ \wedge\, \mathrm{wp}(S, \mathsf{true}) \end{array}\right) \Rightarrow \mathrm{wp}(S; S[\vec{x}/\vec{x}'], d_{\mathsf{Out}}(\vec{x}_{\mathsf{o}}, \vec{x}'_{\mathsf{o}}) \leq Y)$$

$$\left(\begin{array}{l} \mathsf{PIntrst} \wedge \mathsf{StdIn} \\ \wedge\, \mathsf{PIntrst}[\vec{x}/\vec{x}'] \\ \wedge\, f(d_{\mathsf{i}}(\vec{x}_{\mathsf{i}}, \vec{x}'_{\mathsf{i}})) = Y \\ \wedge\, \mathrm{wp}(S[\vec{x}/\vec{x}'], \mathsf{true}) \end{array}\right) \Rightarrow \mathrm{wp}(S[\vec{x}/\vec{x}']; S, d_{\mathsf{Out}}(\vec{x}_{\mathsf{o}}, \vec{x}'_{\mathsf{o}}) \leq Y)$$

CONICET UNC UNIVERSITÄT DES SAARLANDES

Reactive Systems

- A program is interpreted as a function $S : \mathsf{Param} \to \mathsf{In}^\omega \to 2^{(\mathsf{Out}^\omega)}$
- and the set of standard inputs as a language $\mathsf{StdIn} \subseteq \mathsf{In}^\omega$

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$
and $\mathsf{i} \in \mathsf{In}^\omega \cap \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Reactive Systems

- Distances run on *finite* words:

$$d_{\mathsf{In}} : (\mathsf{In}^* \times \mathsf{In}^*) \to \mathbb{R}_{\geq 0} \quad \text{and} \quad d_{\mathsf{Out}} : (\mathsf{Out}^* \times \mathsf{Out}^*) \to \mathbb{R}_{\geq 0}$$

S is *robustly clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}^\omega$, if $\mathsf{i} \in \mathsf{StdIn}$, for all $k \geq 0$,

$$(\forall j \leq k : d_{\mathsf{In}}(\mathsf{i}[..j], \mathsf{i}'[..j]) \leq \kappa_{\mathsf{i}})$$

$$\Rightarrow \mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i})[..k], S(\mathsf{p}')(\mathsf{i}')[..k]) \leq \kappa_{\mathsf{o}}$$

Reactive Systems

- Distances run on *finite* words:

$$d_{\mathsf{In}} : (\mathsf{In}^* \times \mathsf{In}^*) \to \mathbb{R}_{\geq 0} \quad \text{and} \quad d_{\mathsf{Out}} : (\mathsf{Out}^* \times \mathsf{Out}^*) \to \mathbb{R}_{\geq 0}$$

S is *f-clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}^\omega$,
if $\mathsf{i} \in \mathsf{StdIn}$, for all $k \geq 0$,

$$\mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i})[..k], S(\mathsf{p}')(\mathsf{i}')[..k]) \leq f(d_{\mathsf{In}}(\mathsf{i}[..k], \mathsf{i}'[..k]))$$

CONICET

UNC

UNIVERSITÄT DES SAARLANDES

Analysis with HyperLTL

- S is *clean* iff it satisfies

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\ (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \\ \rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_2'} \wedge \mathsf{o}_{\pi_1} = \mathsf{o}_{\pi_2'})\right)$$

Analysis with HyperLTL

- S is *clean* iff it satisfies

Like LTL but adds quantification on traces

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\ (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_2'} \wedge \mathsf{o}_{\pi_1} = \mathsf{o}_{\pi_2'})\right)$$

Analysi… LTL

- S is *clean* iff it satisfies

PIntrs: a propositional formula identifying the parameter of interests

StdIn: an LTL formula identifying the traces with standard input sequences

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\ (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_2'} \wedge \mathsf{o}_{\pi_1} = \mathsf{o}_{\pi_2'})\right)$$

$\bigwedge_{a \in \mathsf{AP_p}} a_{\pi_2} \leftrightarrow a_{\pi_2'}$

$\bigwedge_{a \in \mathsf{AP_i}} a_{\pi_1} \leftrightarrow a_{\pi_2'}$

$\bigwedge_{a \in \mathsf{AP_o}} a_{\pi_1} \leftrightarrow a_{\pi_2'}$

CONICET

UNC

UNIVERSITÄT DES SAARLANDES

Analysis with HyperLTL

- S is *clean* iff it satisfies

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\ (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_2'} \wedge \mathsf{o}_{\pi_1} = \mathsf{o}_{\pi_2'})\right)$$

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$

and $\mathsf{i} \in \mathsf{In}^\omega \cap \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

Analysis with HyperLTL

- S is *clean* iff it satisfies

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\ (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_2'} \wedge \mathsf{o}_{\pi_1} = \mathsf{o}_{\pi_2'})\right)$$

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$

and $\mathsf{i} \in \mathsf{In}^\omega \cap \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) \subseteq S(\mathsf{p}')(\mathsf{i})$

Analysis with HyperLTL

- S is *robustly clean* iff it satisfies

$$d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}') = \hat{d}_{\mathsf{In}}(\mathrm{last}(\mathsf{i}), \mathrm{last}(\mathsf{i}'))$$

$$\forall \pi_1 . \forall \pi_2 . \exists \pi_2' .$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \Big(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \big((\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq \kappa_{\mathsf{o}}) \,\mathsf{W}\, (\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'}) > \kappa_{\mathsf{i}}) \big) \Big)$$

Analysis with HyperLTL

- S is *robustly clean* iff it satisfies

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.\; (\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1}) \rightarrow \Big(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \big((\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq \kappa_{\mathsf{o}})\; \mathsf{W}\; (\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'}) > \kappa_{\mathsf{i}})\big)\Big)$$

S is *robustly clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}^\omega$, if $\mathsf{i} \in \mathsf{StdIn}$, for all $k \geq 0$,

$$(\forall j \leq k : d_{\mathsf{In}}(\mathsf{i}[..j], \mathsf{i}'[..j]) \leq \kappa_{\mathsf{i}}) \Rightarrow \mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i})[..k], S(\mathsf{p}')(\mathsf{i}')[..k]) \leq \kappa_{\mathsf{o}}$$

$$\mathcal{H}(d)(A, B) = \max \left\{ \begin{array}{l} \sup_{a \in A} \inf_{b \in B} d(a, b), \\ \sup_{b \in B} \inf_{a \in A} d(a, b) \end{array} \right\}$$

CONICET UNC UNIVERSITÄT DES SAARLANDES

Analysis with HyperLTL

- S is *robustly clean* iff it satisfies

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_2'.$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \Big(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \big((\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq \kappa_{\mathsf{o}})\ \mathsf{W}\ (\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'}) > \kappa_{\mathsf{i}}) \big) \Big)$$

$$\forall \pi_1.\, \forall \pi_2.\, \exists \pi_1'.$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \Big(\mathsf{p}_{\pi_1} = \mathsf{p}_{\pi_1'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_1'}) \wedge \big((\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1'}, \mathsf{o}_{\pi_2}) \leq \kappa_{\mathsf{o}})\ \mathsf{W}\ (\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1'}, \mathsf{i}_{\pi_2}) > \kappa_{\mathsf{i}}) \big) \Big)$$

CONICET UNC UNIVERSITÄT DES SAARLANDES

Analysis with HyperLTL

- S is *f-clean* iff it satisfies

$$\forall \pi_1 . \forall \pi_2 . \exists \pi_2' .$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \mathsf{G}\left(\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq f(\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'})) \right) \right)$$

S is *f-clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$, whenever $\mathsf{i} \in \mathsf{StdIn}$,

$$\mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i}), S(\mathsf{p}')(\mathsf{i}')) \leq f(d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}'))$$

$$\mathcal{H}(d)(A, B) = \max \left\{ \begin{array}{l} \sup_{a \in A} \inf_{b \in B} d(a, b), \\ \sup_{b \in B} \inf_{a \in A} d(a, b) \end{array} \right\}$$

CONICET UNC UNIVERSITÄT DES SAARLANDES

Analysis with HyperLTL

- S is *f-clean* iff it satisfies

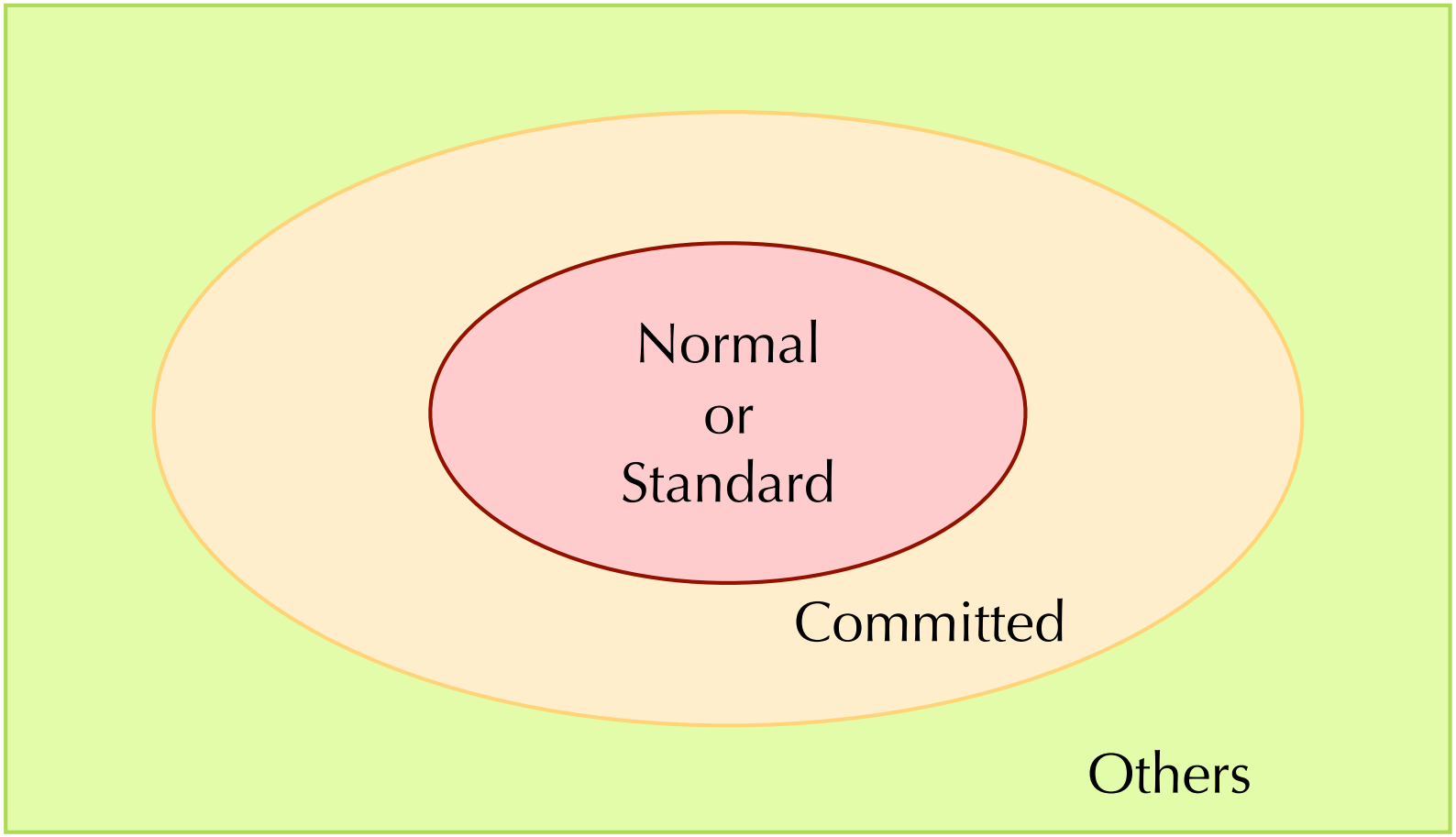
$$\forall \pi_1 . \forall \pi_2 . \exists \pi_2' .$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \left(\mathsf{p}_{\pi_2} = \mathsf{p}_{\pi_2'} \wedge \mathsf{G}(\mathsf{i}_{\pi_2} = \mathsf{i}_{\pi_2'}) \wedge \mathsf{G}\left(\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1}, \mathsf{o}_{\pi_2'}) \leq f(\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1}, \mathsf{i}_{\pi_2'})) \right) \right)$$

$$\forall \pi_1 . \forall \pi_2 . \exists \pi_1' .$$
$$(\mathsf{PIntrs}_{\pi_1} \wedge \mathsf{PIntrs}_{\pi_2} \wedge \mathsf{StdIn}_{\pi_1})$$
$$\rightarrow \left(\mathsf{p}_{\pi_1} = \mathsf{p}_{\pi_1'} \wedge \mathsf{G}(\mathsf{i}_{\pi_1} = \mathsf{i}_{\pi_1'}) \wedge \mathsf{G}\left(\hat{d}_{\mathsf{Out}}(\mathsf{o}_{\pi_1'}, \mathsf{o}_{\pi_2}) \leq f(\hat{d}_{\mathsf{In}}(\mathsf{i}_{\pi_1'}, \mathsf{i}_{\pi_2})) \right) \right)$$

Model checked a toy version of the emission control case study

CONICET

A general contract



Inputs

A general contract

S is *clean* if for all $\mathsf{p}, \mathsf{p}' \in \mathsf{PIntrst}$ and $\mathsf{i}, \mathsf{i}' \in \mathsf{In}$,

1. if $\mathsf{i} \in \mathsf{StdIn}$, $S(\mathsf{p})(\mathsf{i}) = S(\mathsf{p}')(\mathsf{i})$

2. if $\mathsf{i} \in \mathsf{StdIn}$ and $\mathsf{i}' \in \mathsf{Comm}$

$$\mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i}), S(\mathsf{p}')(\mathsf{i}')) \leq f(d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}'))$$

3. if $\mathsf{i}' \notin \mathsf{StdIn} \cup \mathsf{Comm}$, then for all ϵ, exists δ s.t. for all $\mathsf{i} \in \mathsf{In}$,

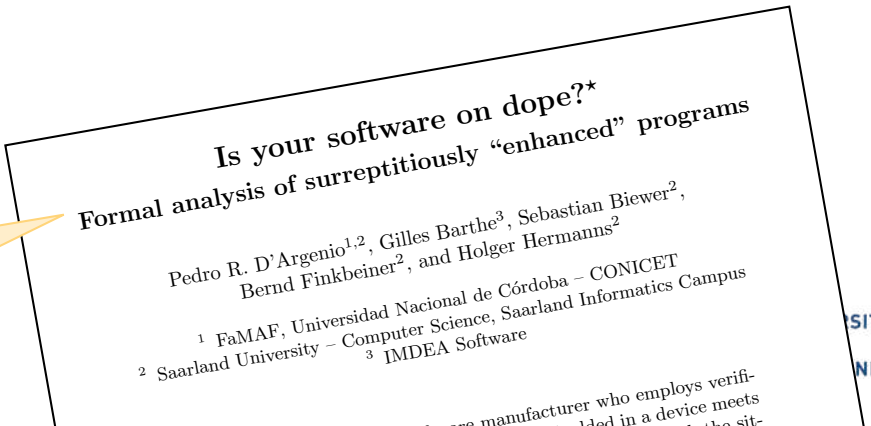
$$d_{\mathsf{In}}(\mathsf{i}, \mathsf{i}') < \delta \;\Rightarrow\; \mathcal{H}(d_{\mathsf{Out}})(S(\mathsf{p})(\mathsf{i}), S(\mathsf{p}')(\mathsf{i}')) < \epsilon$$

Concluding remark

- We discussed what is software doping
- and motivate it with concrete examples
- Several formal characterizations of software doping
- They can be analyzed using self-composition

  (for deterministic programs)
- We also studied characterizations for reactive (non-deterministic) systems

ESOP 2017
to appear



Is your software on dope?*

Formal analysis of surreptitiously "enhanced" programs

Pedro R. D'Argenio[1,2], Gilles Barthe[3], Sebastian Biewer[2], Bernd Finkbeiner[2], and Holger Hermanns[2]

[1] FaMAF, Universidad Nacional de Córdoba – CONICET
[2] Saarland University – Computer Science, Saarland Informatics Campus
[3] IMDEA Software

... manufacturer who employs verifi-
... ded in a device meets
... the sit-

CONICET

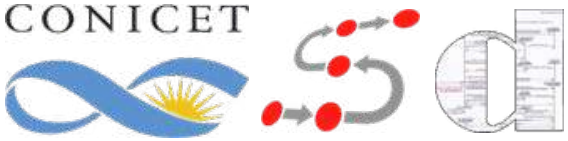
Is your software on dope?

Formal analysis of surreptitiously "enhanced" programs

Gilles Barthe, Sebastian Biewer, Pedro R. D'Argenio,
Bernd Finkbeiner, and Holger Hermanns

IMDEA Software (ES) UN Córdoba – CONICET (AR) Saarland University (DE)

http://www.cs.famaf.unc.edu.ar/~dargenio/



FMT, UT, January 2017