Efficient Data Validation for Geographical Interlocking Systems

Jan Peleska, Niklas Krafczyk
University of Bremen
{peleska,niklas}@uni-bremen.de

Anne E. Haxthausen
Denmark Technical University DTU
aeha@dtu.dk

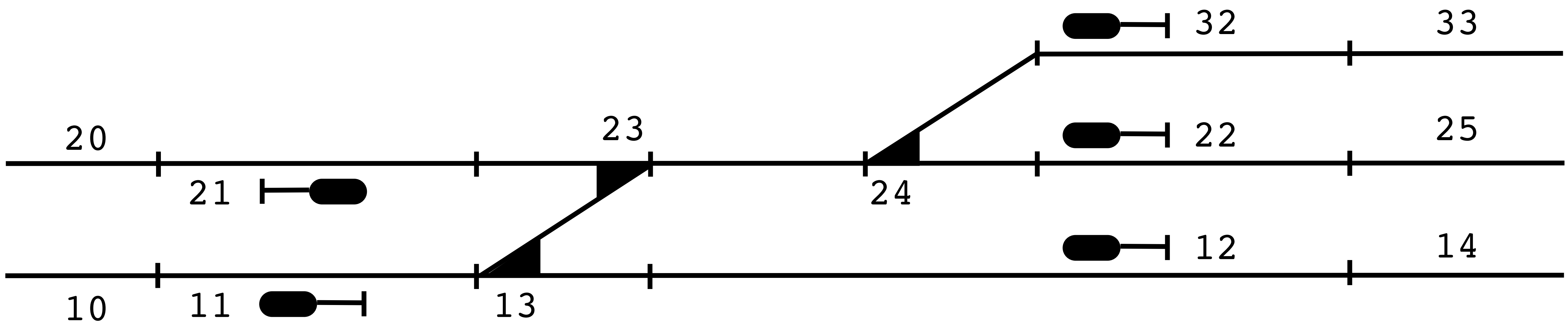
Ralf Pinger
Siemens Mobility GmbH
ralf.pinger@siemens.com

RSSRail 2019, 4 – 6 June 2019, Lille, France

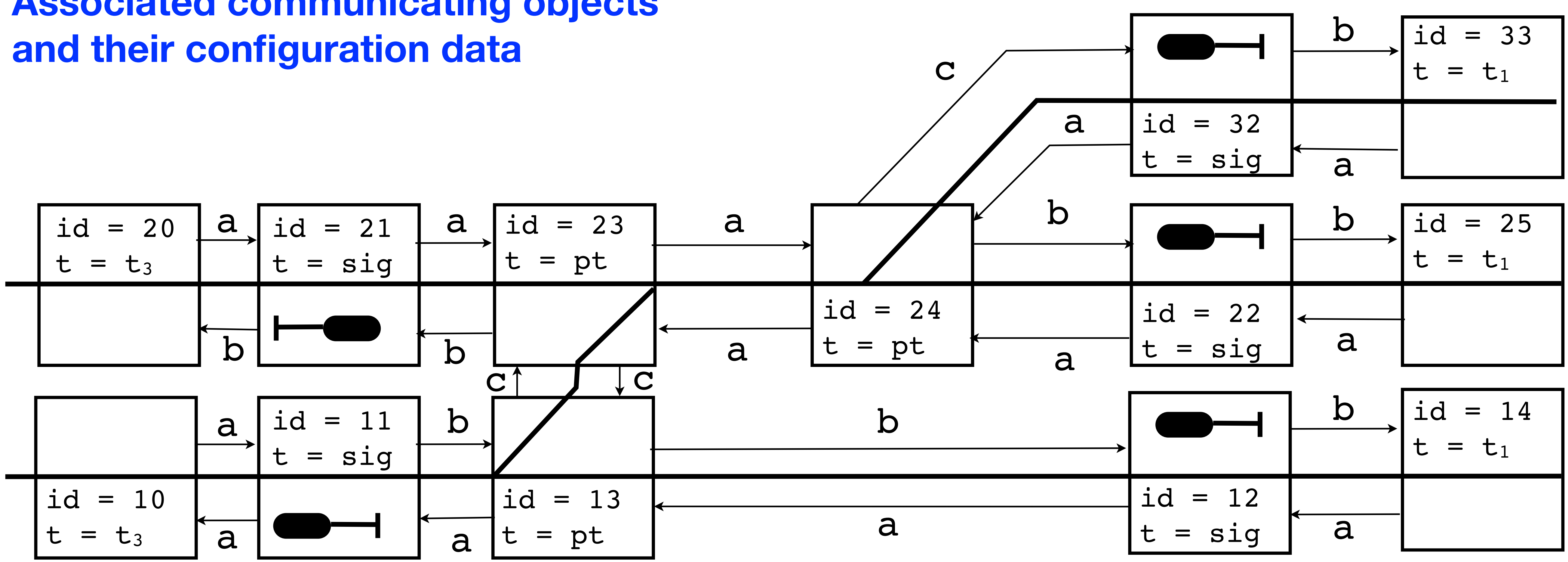
Data Validation

- **Geographical Interlocking System (IXL)**
  - Routes from start point to destination point are dynamically allocated
- **Data validation**
  - Ensure that the IXL configuration data conforms to rules applicable to the track elements involved

Real world – track elements



Associated communicating objects and their configuration data



Main Contributions

- **Data validation is transformed into a property checking problem** using Kripke Structures and Temporal Logic
- Checking problem is **over-approximated using CTL**
- Global track model is decomposed into **directed sub-models**
- This allows for application of **very fast parallelised global model checking algorithms**
- Checking rule violations by means of properties specified in CTL is **very easy to use** and can be based on templates

Related Work

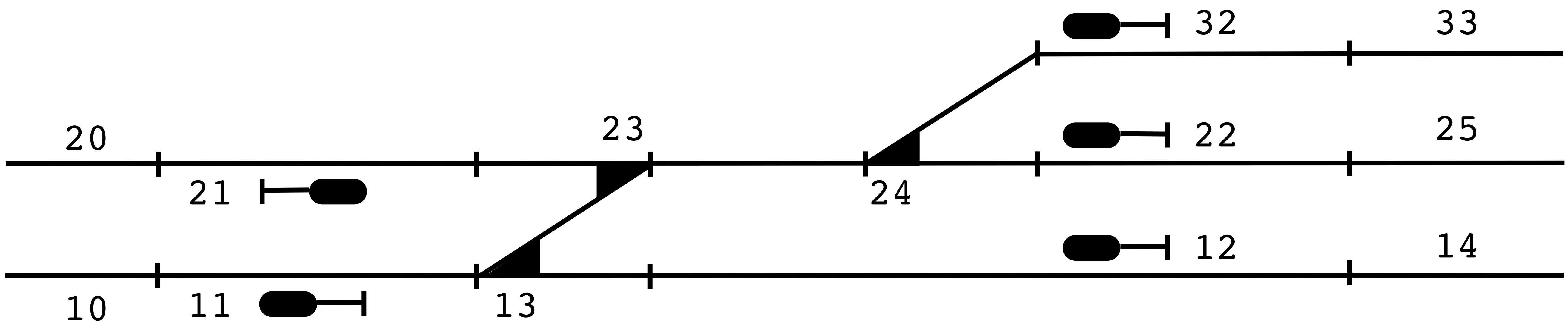
- Data validation can also be automated using, for example, the B-Method and associated tools
  - Badeau, F., Doche-Petit, M.: Formal data validation with Event-B. arXiv:1210.7039 [cs], October 2012
  - Fredj, M., Leger, S., Feliachi, A., Ordioni, J.: OVADO. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. LNCS, vol. 10598, pp. 87–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4 6
  - Hansen, D., Schneider, D., Leuschel, M.: Using B and ProB for data validation projects. In: Butler, M., Schewe, K.-D., Mashkoor, A., Biro, M. (eds.) ABZ 2016. LNCS, vol. 9675, pp. 167–182. Springer, Cham (2016). https://doi.org/10.1007/ 978-3-319-33600-8 10
  - Keming, W., Zheng, W., Chuandong, Z.: Formal modeling and data validation of general railway interlocking system. WIT Trans. Built Environ. 181, 527–538 (2018)

Overview

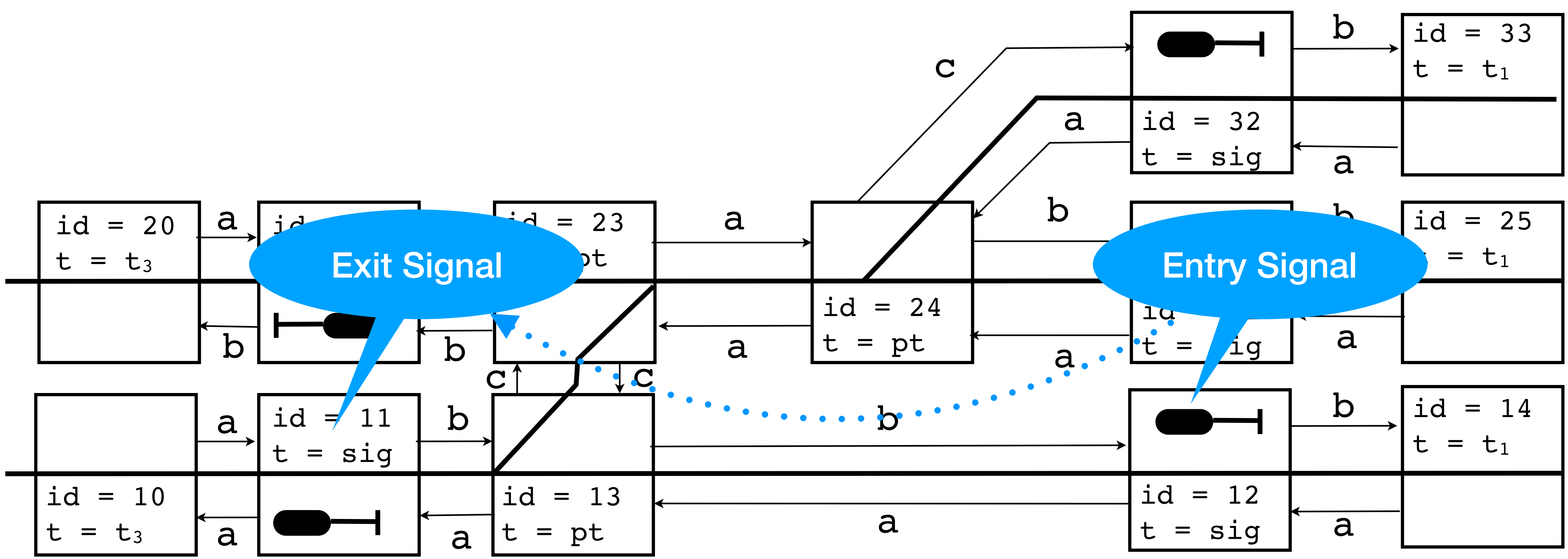
- Data validation rules – more details
- IXL Configurations as Kripke Structures – sub-models
- Rule violation representation in LTL
- From LTL to CTL – global model checking
- Evaluation & Conclusion

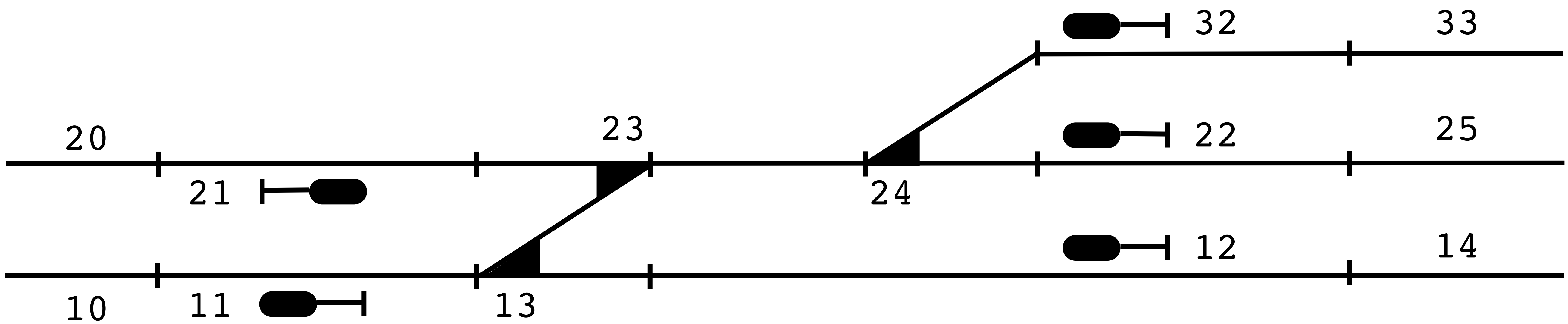
Data Validation Rules

- Data validation rules specify
  - **Constraints about parameter values** for element instances, depending on their location in the network
  - **Constraints about element types**, depending on the sequences of element instances

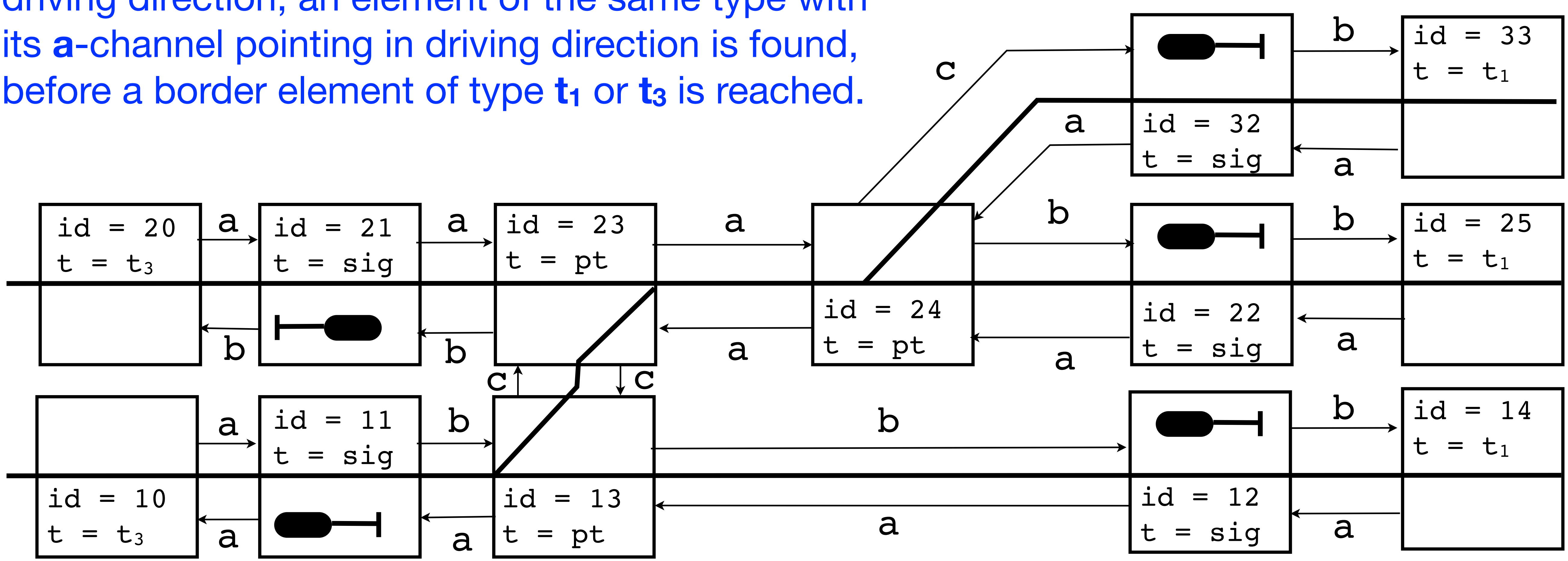


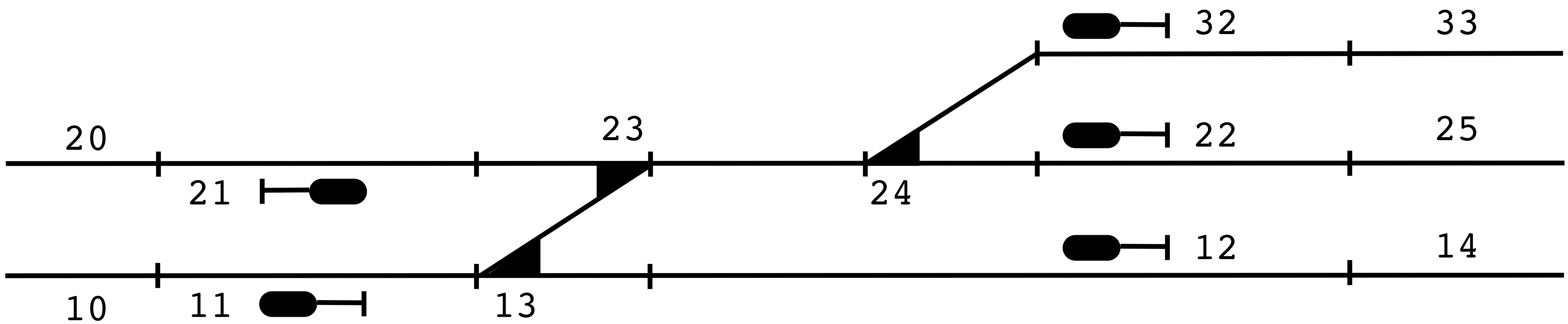
Rule 1. Every **entry signal** is associated with a corresponding **exit signal**



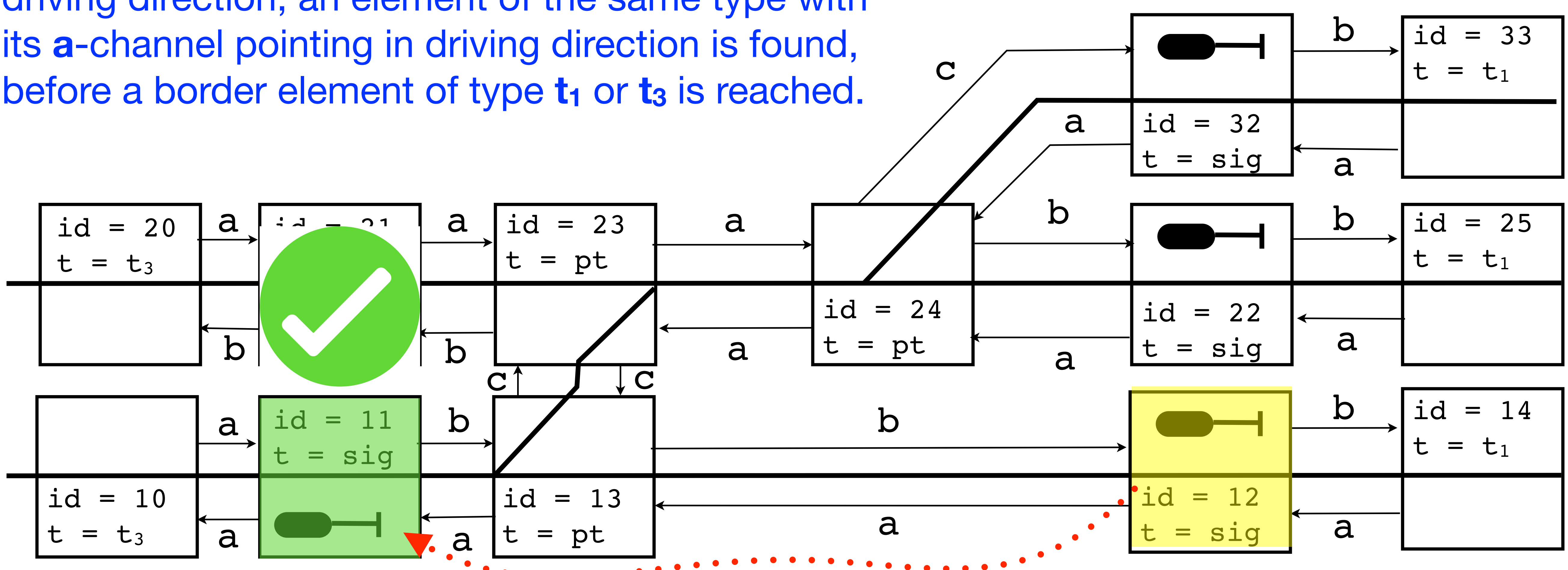


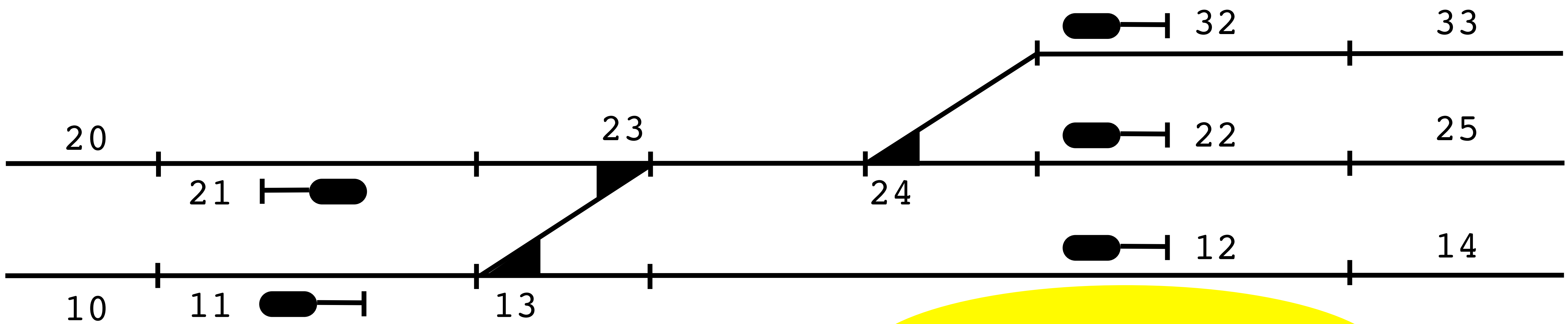
Rule 1 (technical re-formulation). From channel **a** of an element of type **sig** at block entry, pointing in driving direction, an element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type **t_1** or **t_3** is reached.



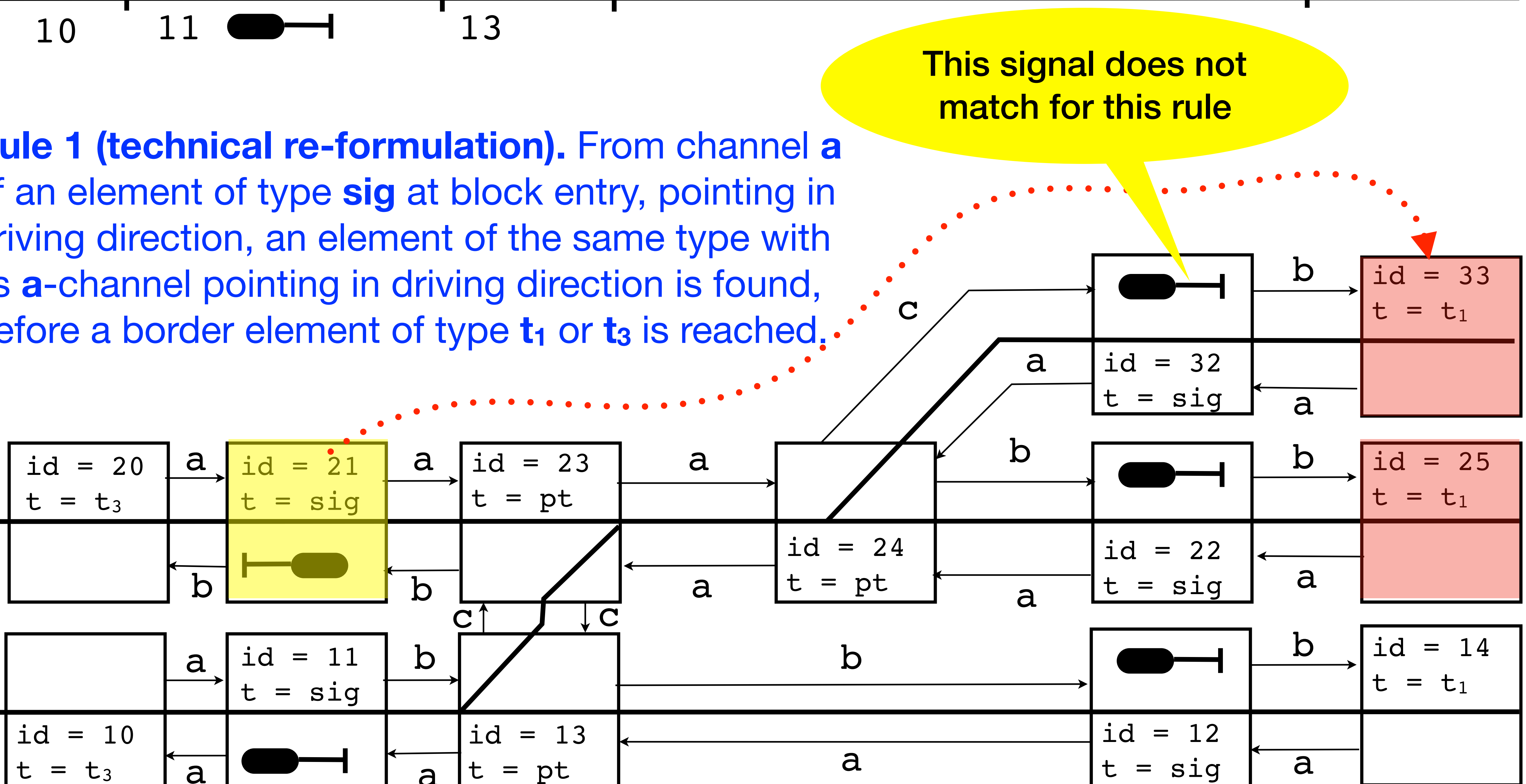


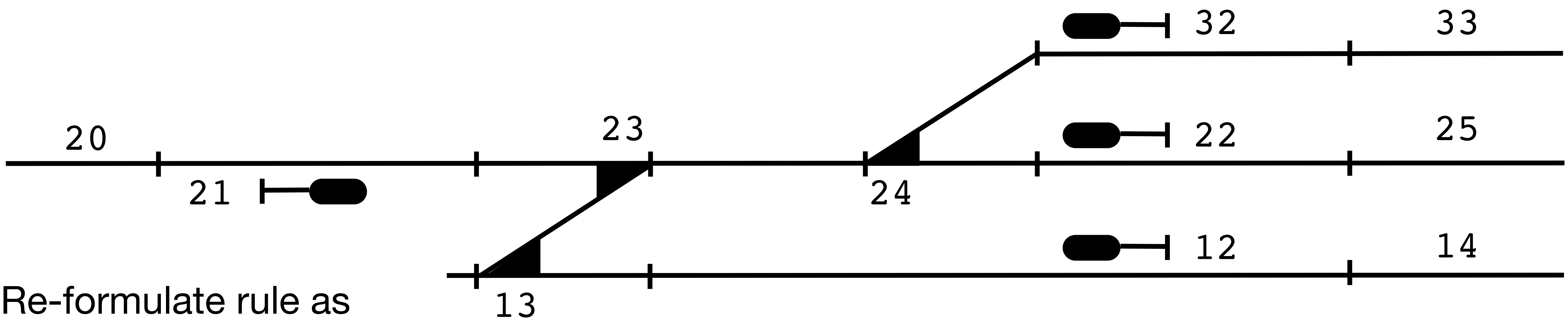
Rule 1 (technical re-formulation). From channel **a** of an element of type **sig** at block entry, pointing in driving direction, an element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type $\mathbf{t_1}$ or $\mathbf{t_3}$ is reached.





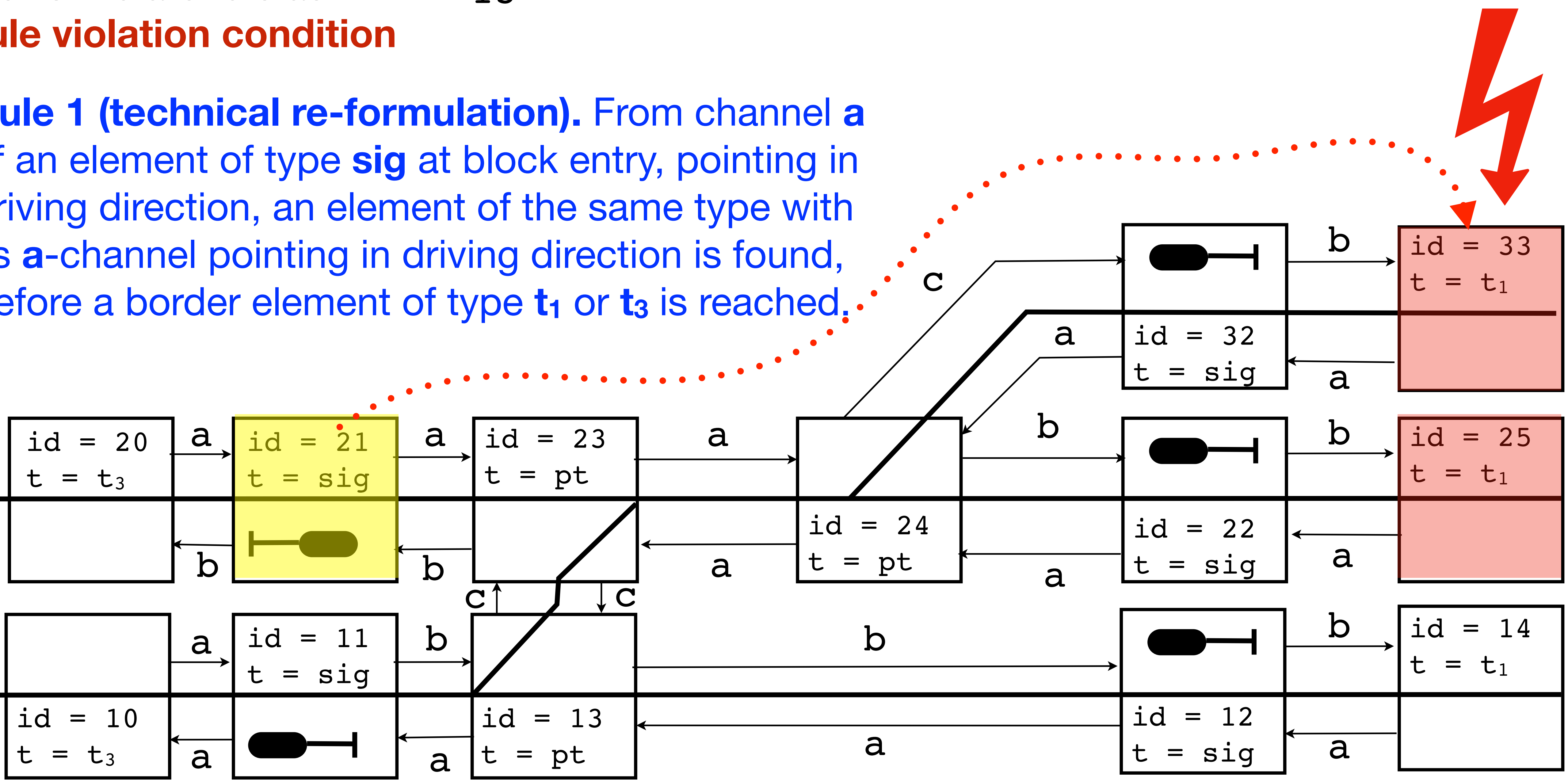
Rule 1 (technical re-formulation). From channel **a** of an element of type **sig** at block entry, pointing in driving direction, an element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type $\mathbf{t_1}$ or $\mathbf{t_3}$ is reached.

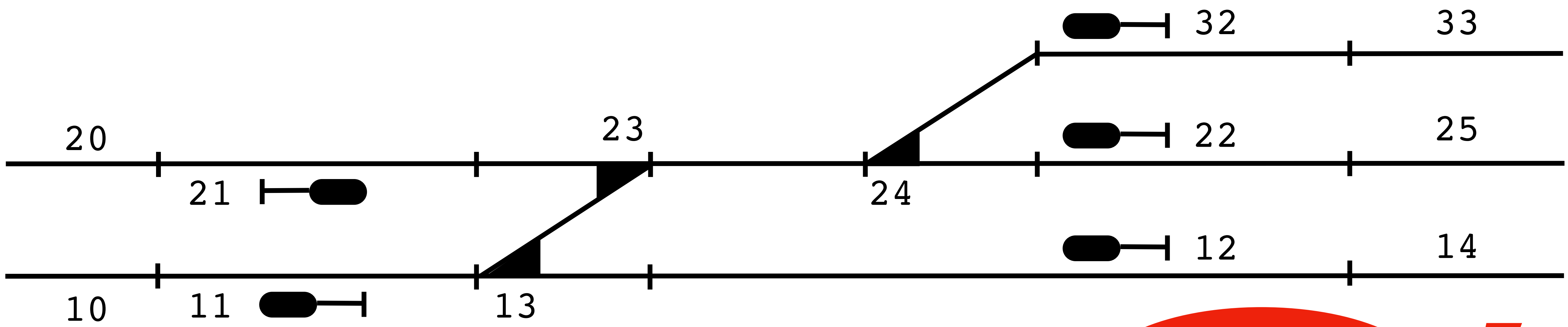




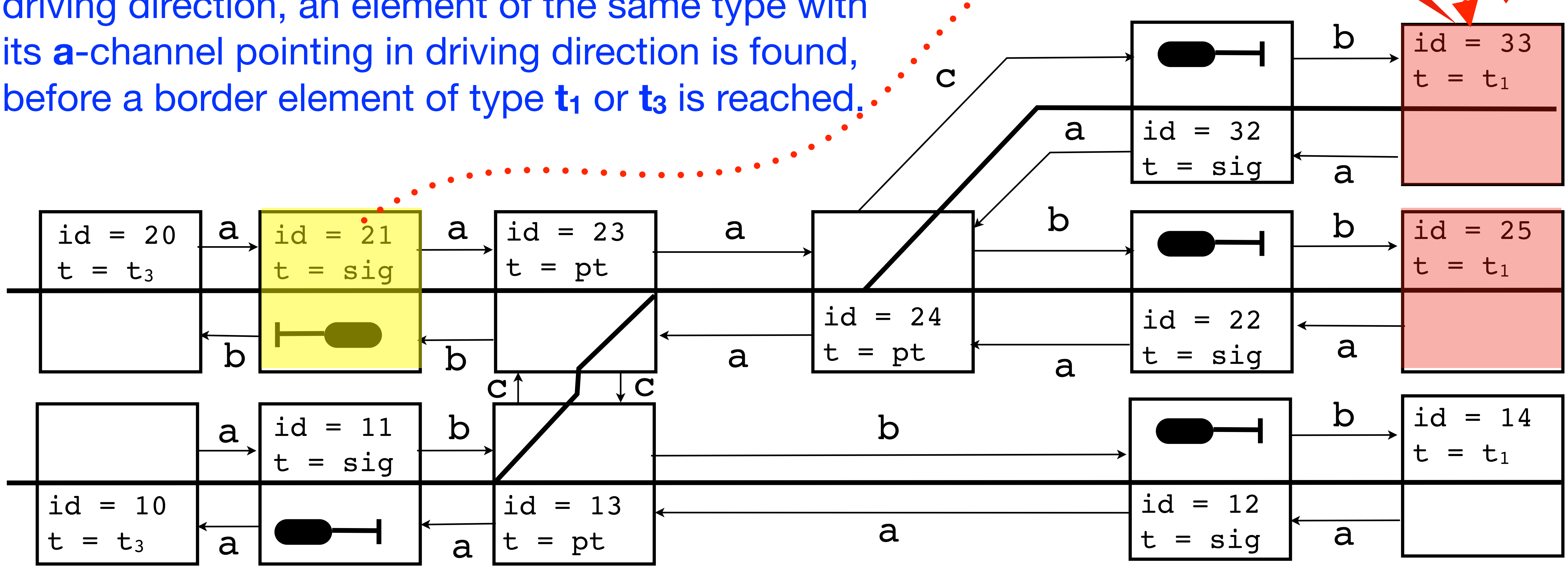
Re-formulate rule as
rule violation condition

Rule 1 (technical re-formulation). From channel **a** of an element of type **sig** at block entry, pointing in driving direction, an element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type $\mathbf{t_1}$ or $\mathbf{t_3}$ is reached.





Rule 1 (technical re-formulation). From channel **a** of an element of type **sig** at block entry, pointing in driving direction, an element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type **t_1** or **t_3** is reached.



IXL Configurations as Kripke Structures

- **Kripke Structure**

$$
\begin{aligned}
K &= (S, S_0, R, L, AP) \\
S &= \text{state space} \\
S_0 \subseteq S &= \text{initial states} \\
R \subseteq S \times S &= \text{transition relation} \\
L : S \to 2^{AP} &= \text{Mapping from states to sets of} \\
& \quad \text{valid atomic propositions from } AP
\end{aligned}
$$

IXL sub-model configurations encoded as Kripke Structure

Sub-model. Sub-graph of element instances from one entry element to all exit elements reachable in driving direction

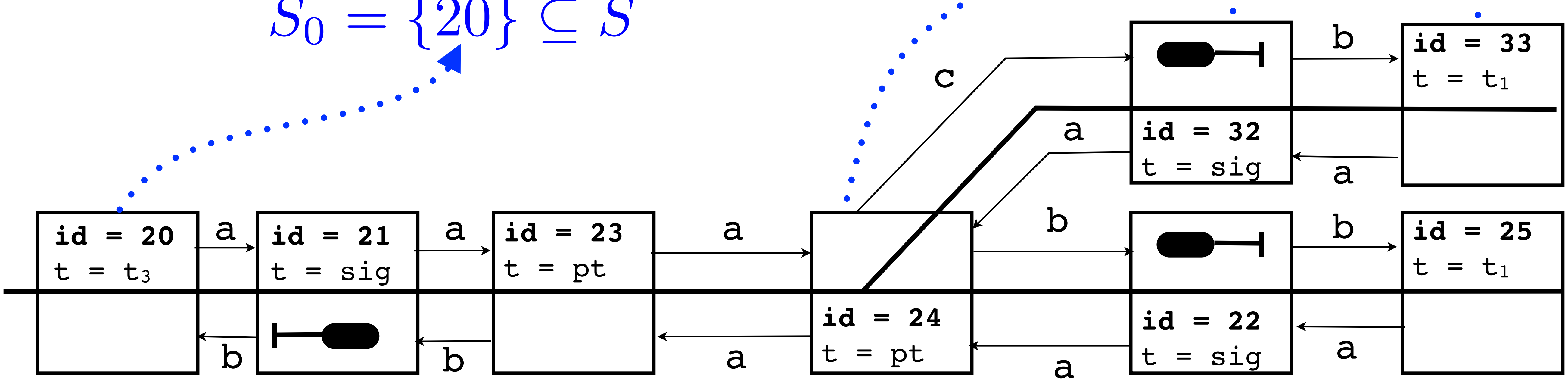
$$K = (S, S_0, R, L, AP)$$

States are element instances, identified by their id

$$S = \{20, 21, 23, 24, 32, 33, 22, 25\}$$

Initial state is the entry element

$$S_0 = \{20\} \subseteq S$$



Driving direction

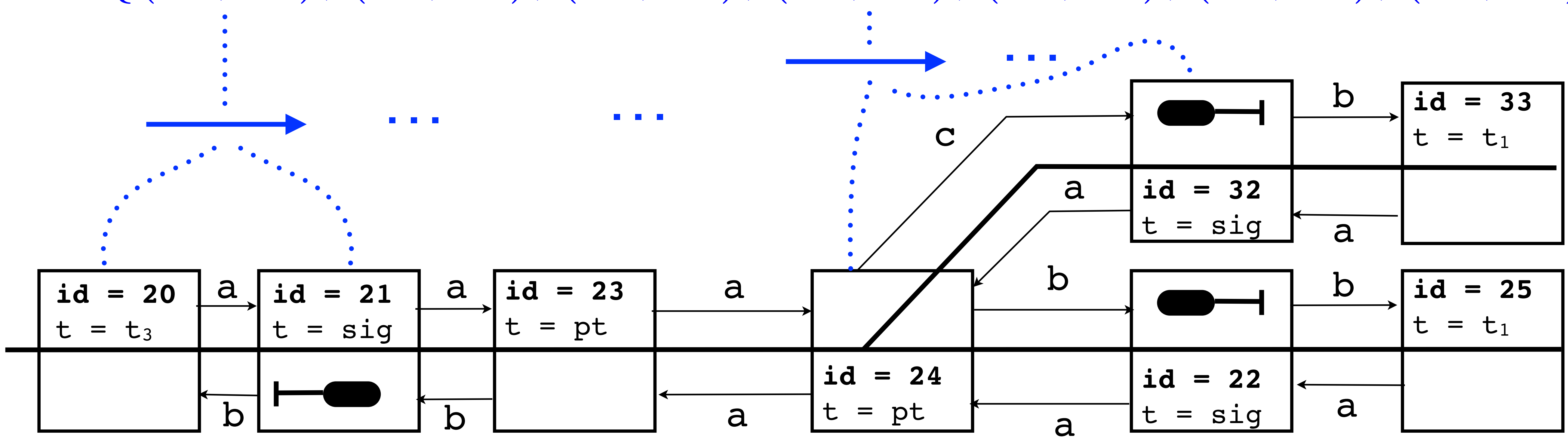
IXL sub-model configurations encoded as Kripke Structure

Sub-model. Sub-graph of element instances from one entry element to all exit elements reachable in driving direction

$$K = (S, S_0, R, L, AP)$$

Transition Relation. Pairs of elements linked by primary channel a, b, c, d in driving direction

$$R = \{(20, 21), (21, 23), (23, 24), (24, 32), (24, 22), (32, 33), (22, 25)\}$$



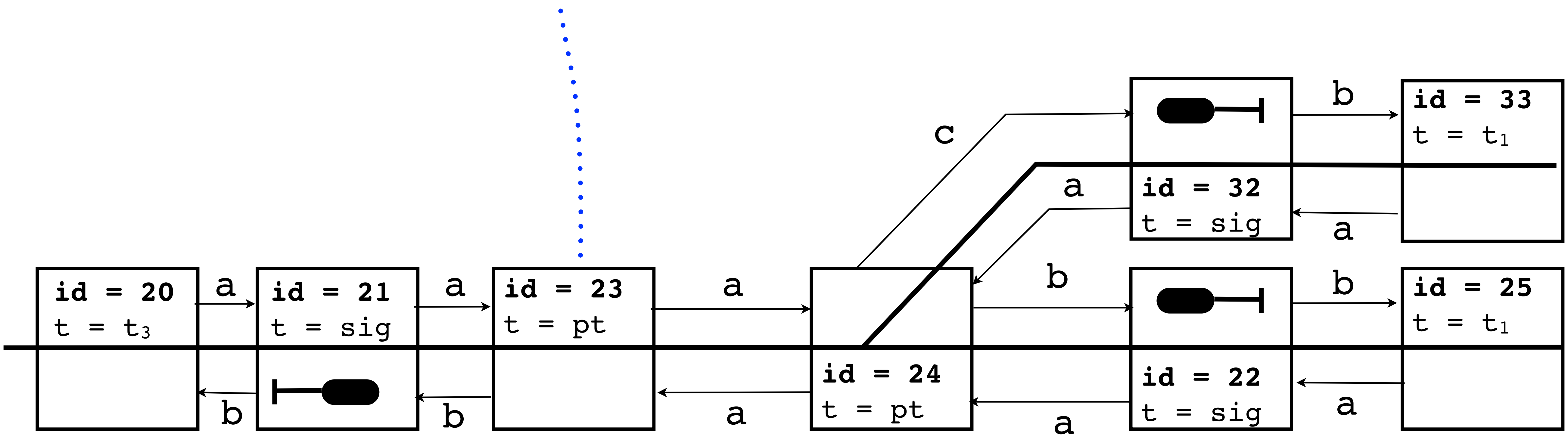
IXL sub-model configurations encoded as Kripke Structure

Sub-model. Sub-graph of element instances from one entry element to all exit elements reachable in driving direction

$$K = (S, S_0, R, L, AP)$$

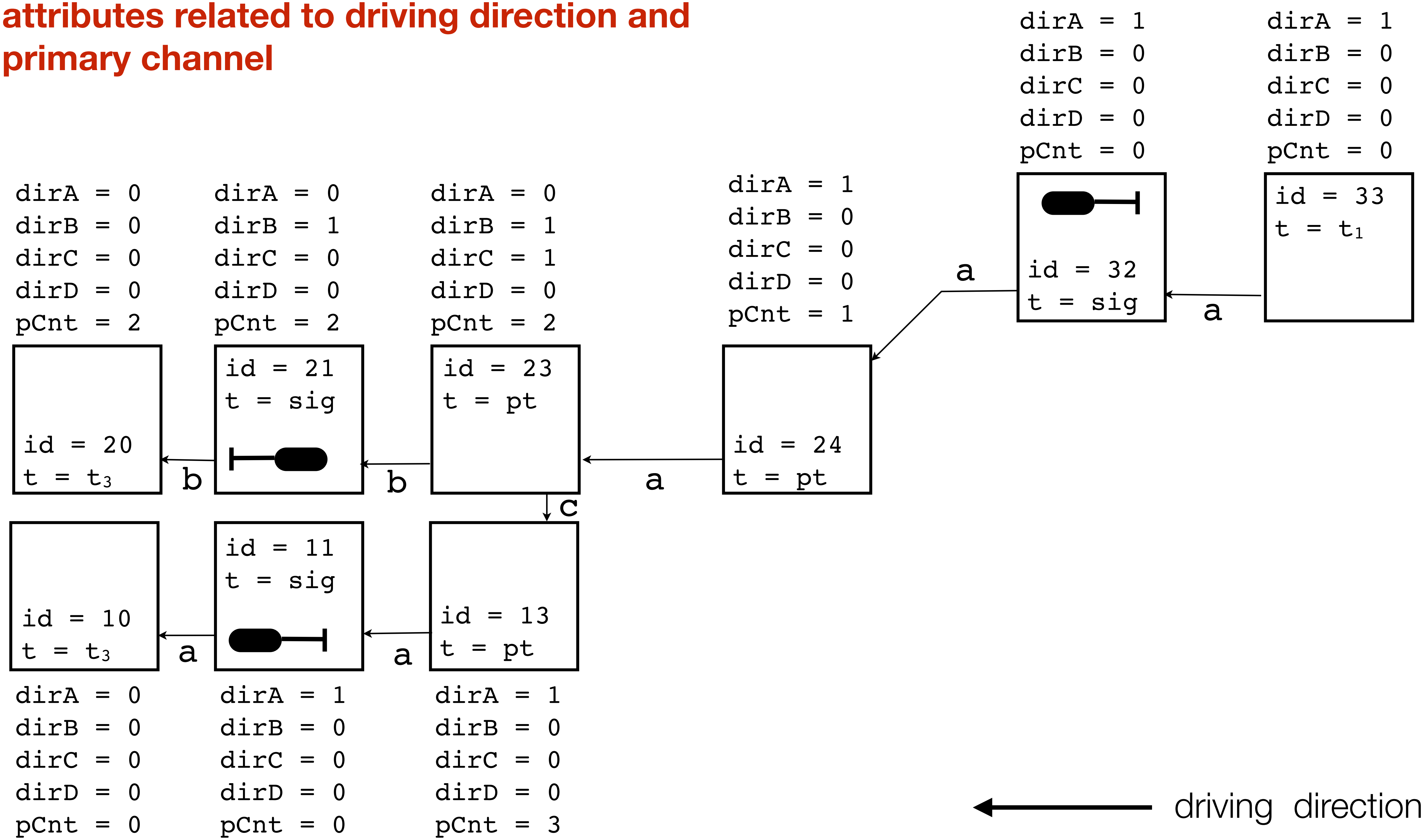
Labelling function. Maps states to set of propositions specifying the parameter values

$$L(23) = \{id = 23, t = pt, \ldots\}$$



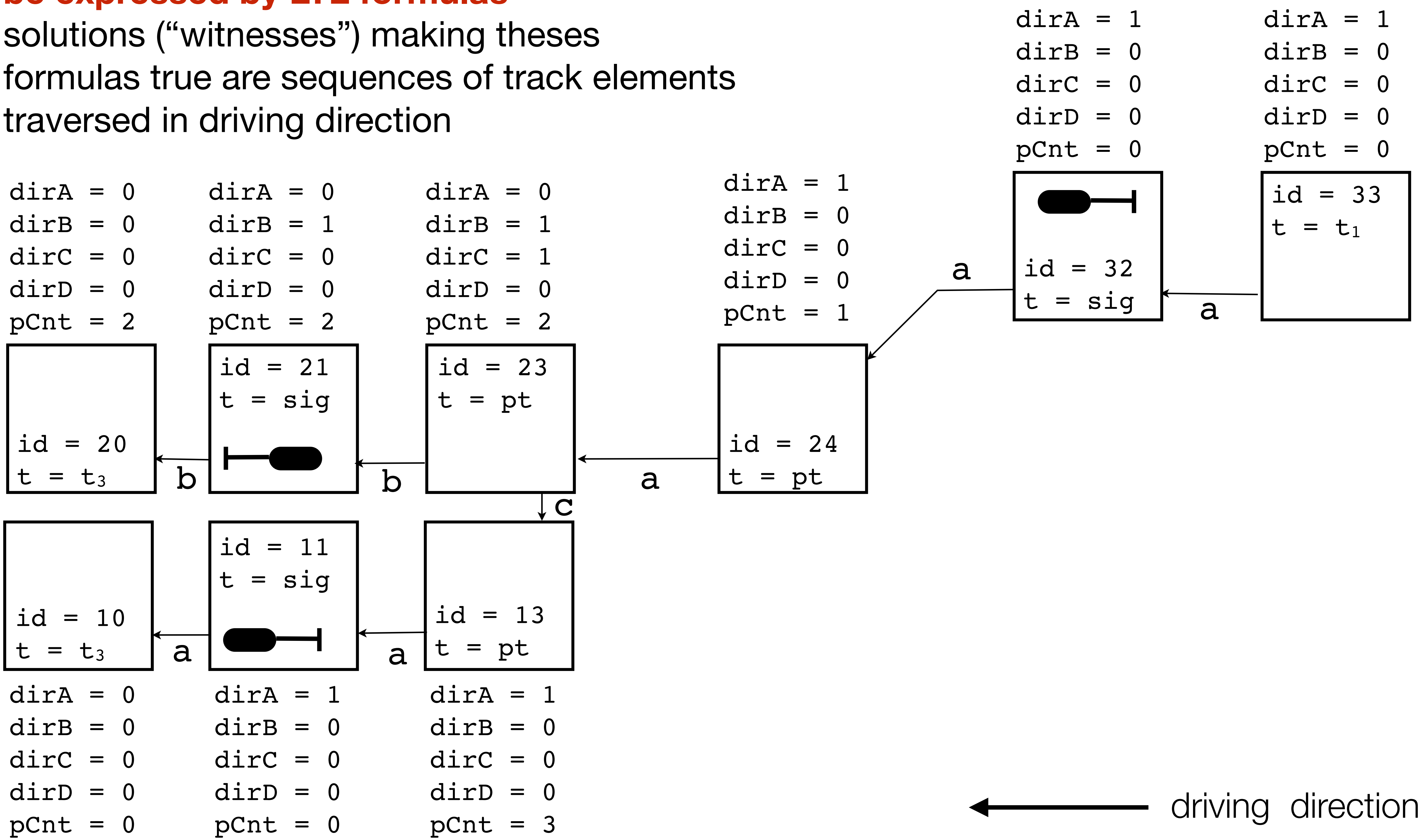
IXL sub-model configurations encoded as Kripke Structure

Sub-model states are equipped with **additional attributes related to driving direction and primary channel**



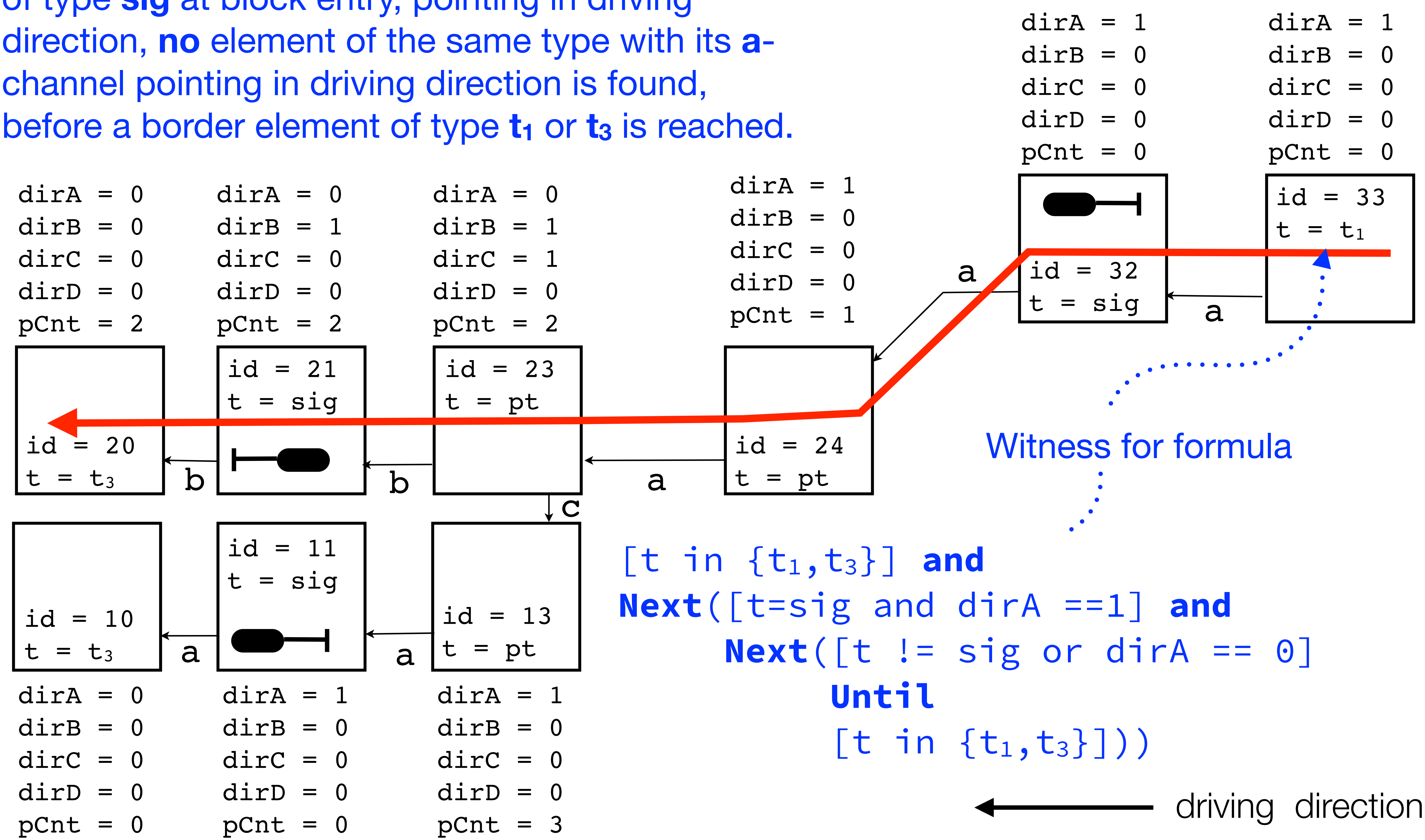
Rule Representation in LTL

On sub-models, **rule violations may be expressed by LTL formulas** – solutions (“witnesses”) making theses formulas true are sequences of track elements traversed in driving direction



Rule Violation Representation in LTL

Violation of Rule 1. From channel **a** of an element of type **sig** at block entry, pointing in driving direction, **no** element of the same type with its **a**-channel pointing in driving direction is found, before a border element of type **t_1** or **t_3** is reached.




```
[t in {t₁,t₃}] and
Next([t=sig and dirA ==1] and
     Next([t != sig or dirA == 0]
          Until
          [t in {t₁,t₃}]))
```

Which LTL Subset is Needed?

- Data validation rules are **safety formulas**
- Their violation can be detected on a finite path prefix

Theorem. Rule violations can be expressed by first-order expressions composed by operators **and**, **or**, **Next**, **Until** alone.

Proof is based on well-known result from

Sistla, A.P.: Safety, liveness and fairness in temporal logic. Formal Aspects Comput. 6(5), 495–511 (1994). https://doi.org/10.1007/BF01211865

From LTL to CTL

- LTL model checking is PSPACE complete

Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733–749 (1985). https://doi.org/10.1145/3828.3837

- CTL model checking has running time $O(|f| \cdot (|S| + |R|))$

Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)

- **Therefore, CTL model checking is generally much faster than LTL model checking**

From LTL to CTL

Translation from required LTL subset to CTL

$\Phi(f) = f \quad \text{for first-order formulas } f$

$\Phi(\psi_1 \wedge \psi_2) = \Phi(\psi_1) \wedge \Phi(\psi_2)$

$\Phi(\psi_1 \vee \psi_2) = \Phi(\psi_1) \vee \Phi(\psi_2)$

$\Phi(\mathbf{Next}\psi_1) = \underline{\mathbf{Exists}}\big(\mathbf{Next}(\Phi(\psi_1))\big)$

$\Phi(\psi_1 \mathbf{Until}\ \psi_2) = \underline{\mathbf{Exists}}(\Phi(\psi_1)\mathbf{Until}\ \Phi(\psi_2))$

In CTL, this states the existence of a path through the Kripke Structure

Theorem. Let π be any path and ψ an LTL formula specifying a safety violation on π. Let K be a Kripke structure over state space S containing π as a computation. Then

$$\pi \models_{\mathrm{LTL}} \psi \quad \text{implies} \quad K \models_{\mathrm{CTL}} \Phi(\psi).$$

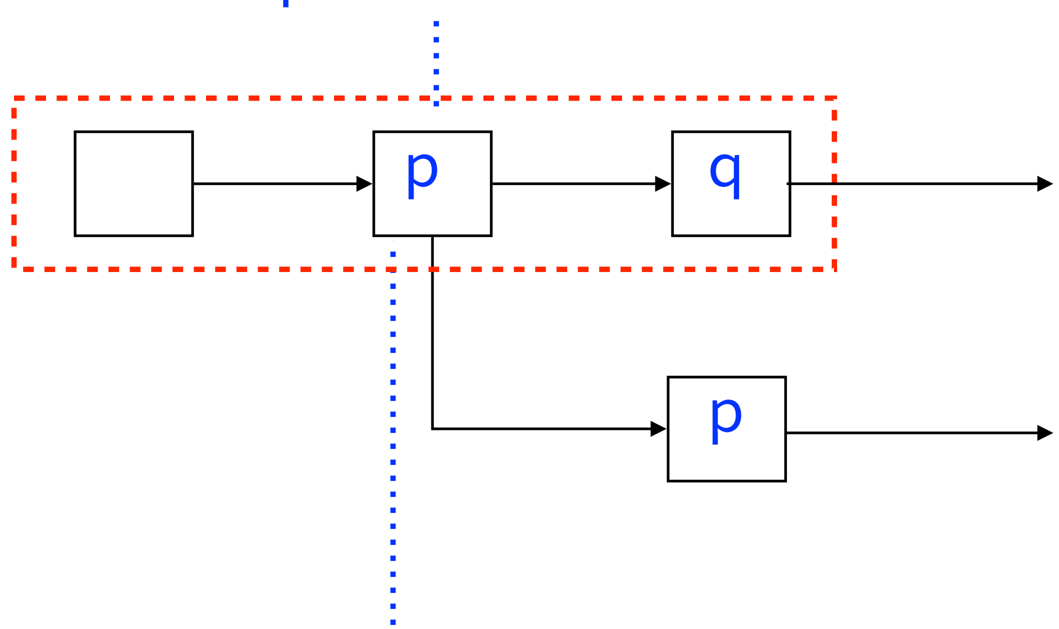
Mapping from LTL to CTL is Over-Approximation

Example of an LTL formula where the CTL translation produces a false witness

(Next p**) Until** q $\xrightarrow{\Phi}$ **Exists((Exists(Next** p**)) Until** q**)**

ψ $\Phi(\psi)$

Next p does not hold



Exists(Next p**)** holds

This path is a witness for $\Phi(\psi)$ in CTL, but it is not a witness for ψ in LTL

False Witnesses can be Detected

The CTL witness can be checked whether it's also an LTL witness

This check can be performed again with running time $O(|f| \cdot (|S| + |R|))$

Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)

False Witnesses can be Avoided

- Instead of checking the whole sub-model,
  - decompose it further and check only linear paths between two border elements (start and end)
- Linear paths are trivial Kripke Structures, and CTL model checking is equivalent to LTL model checking

Evaluation

- Sub-models may be checked concurrently on multi-core systems
- Even for the most complex configurations available, all rules could be checked within less than 10s.
- No false witnesses where ever encountered for the rule violation formulas provided by Siemens

Conclusion

- Data validation for geographic IXLs can be encoded as an LTL model checking problem
- Queries – i.e. formulas detecting rule violations – are easy to specify
- Model checking is fast since
  - global CTL model checking can be used
  - checking problem can be parallelised
- End users prefer global model checking tool to previous bounded model checker, since the latter could not prove the absence of configuration errors
- Current checker is far more effective than previous verification programs working with hard-coded rules