A Formal Approach to Designing Autonomous Systems:

from Intelligent Transport Systems to Autonomous Robots

Fabrice Kordon[1] Laure Petrucci[2]

[1]LIP6, University Paris 6
Paris, France

[2]LIPN, University Paris 13
Villetaneuse, France

Motivation

Specifics critical complex systems

- intrinsically highly distributed systems
- ensure correct behaviour
- real-time properties: durations, timeouts, . . .
- real-space aspects: safety distances, . . .

Use of formal methods

- describe the problem
- verify that expected properties are satisfied
- simulation not sufficient
- formal methods are mathematically sound
- provide automated tools

Motivation

Specifics critical complex systems

- intrinsically highly distributed systems
- ensure correct behaviour
- real-time properties: durations, timeouts, . . .
- real-space aspects: safety distances, . . .

Use of formal methods

- describe the problem
- verify that expected properties are satisfied
- simulation not sufficient
- formal methods are mathematically sound
- provide automated tools

Outline

Intelligent Transport Systems

Characteristics

- highly critical: failures may lead to fatal accidents
- involve a significant number of partners
- cooperation between partners must be both efficient and secure

Partners involved

- road operators
- infrastructure
- vehicles with or without embedded equipment
- drivers

Intelligent Transport Systems

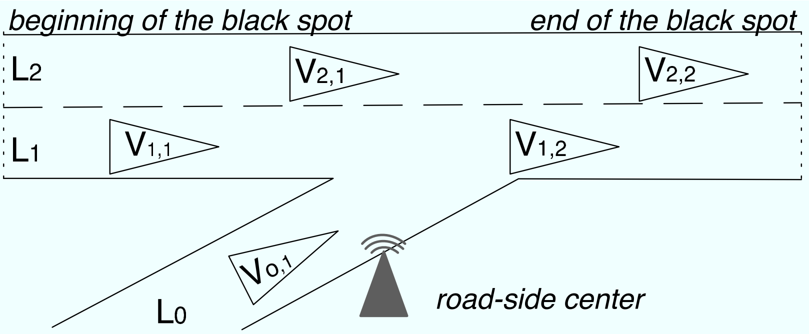
Characteristics

- highly critical: failures may lead to fatal accidents
- involve a significant number of partners
- cooperation between partners must be both efficient and secure

Partners involved

- road operators
- infrastructure
- vehicles with or without embedded equipment
- drivers

ITS example: safe insertion in a motorway



Black-spot functionning

Requirements

- minimum distance between vehicles within a same lane
- vehicles in the entrance lane must eventually enter the motorway
- vehicles should not stop

A cyclic functionning

1. vehicles get their position
2. they send their position to the infrastructure
3. when the infrastructure has received all positions, it issues commands according to a predefined strategy

Black-spot functionning

Requirements

- minimum distance between vehicles within a same lane
- vehicles in the entrance lane must eventually enter the motorway
- vehicles should not stop

A cyclic functionning

1. vehicles get their position
2. they send their position to the infrastructure
3. when the infrastructure has received all positions, it issues commands according to a predefined strategy

Modelling issues

Requirements

- managing dynamic actors
- modelling physical aspects
- preserving a fair progression of the system: actors perform actions at a similar pace.

Development steps

1. first formal specification
2. qualitative analysis: is the global behaviour correct?
3. refinement including time or space features
4. quantitative analysis: do the envisionned strategies satisfy the physical constraints?

Modelling issues

Requirements

- managing dynamic actors
- modelling physical aspects
- preserving a fair progression of the system: actors perform actions at a similar pace.

Development steps

1. first formal specification
2. qualitative analysis: is the global behaviour correct?
3. refinement including time or space features
4. quantitative analysis: do the envisionned strategies satisfy the physical constraints?

Modelling issues

Requirements

- managing dynamic actors
- modelling physical aspects
- preserving a fair progression of the system: actors perform actions at a similar pace.

Development steps

1. first formal specification
2. qualitative analysis: is the global behaviour correct?
3. refinement including time or space features
4. quantitative analysis: do the envisionned strategies satisfy the physical constraints?

Specification issues

UML

- \+ industrial standard
- \+ structured models
- \- semantics not formal enough
- \- analysis by simulation

Algebraic methods

- \+ formally defined
- \- tools require experienced users

Model-checking

- \+ structured models
- \+ large automated tools support
- \+ exhaustive and efficient analysis techniques
- \- cope with models complexity

Choosing the formalism

Requirements

- capture relevant aspects of the problem
- simple enough
- allow for efficient verification techniques
- tool support

⇒ Symmetric Nets

Choosing the formalism

Requirements

- capture relevant aspects of the problem
- simple enough
- allow for efficient verification techniques
- tool support

⇒ Symmetric Nets

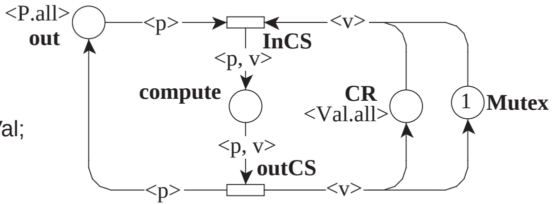
Symmetric nets basic features

SN characteristics

- high-level features
- simple data and functions: enumerated types, intervals, tuples ; predecessor, successor, selector and *broadcast*

Class
  P is 1..**PR**;
  Val is 1..**V**;
Domain
  D is <P,Val>;

Var
  p in P;
  v, v2 in Val;

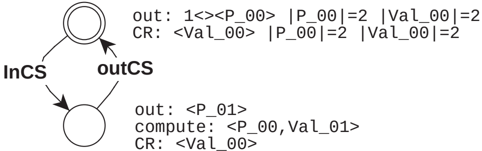


Dedicated analysis techniques

Symbolic reachability graph

- nodes represent a set of states with a similar structure
- based on symmetries computation

⇒ well-suited for ITS which enjoy intrinsic symmetries (e.g. same algorithm in all cars)

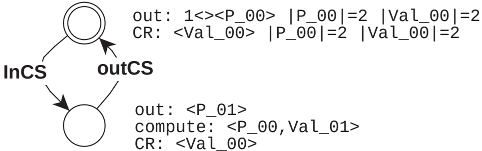


Dedicated analysis techniques

Symbolic reachability graph

- nodes represent a set of states with a similar structure
- based on symmetries computation

⇒ well-suited for ITS which enjoy intrinsic symmetries (e.g. same algorithm in all cars)



Components and abstraction level

Why is structure necessary?

- interaction between components
- communication mechanisms: asynchronous (place fusion) or synchronous (transition fusion)

Advantages

- reusability: try out several models for a single component
- use of modular analysis techniques

Abstraction

- apply a refinement process
- add details step by step until the desired abstraction level is reached
- start with general behaviour and add the real-time and real-space aspects at a later stage

Components and abstraction level

Why is structure necessary?

- interaction between components
- communication mechanisms: asynchronous (place fusion) or synchronous (transition fusion)

Advantages

- reusability: try out several models for a single component
- use of modular analysis techniques

Abstraction

- apply a refinement process
- add details step by step until the desired abstraction level is reached
- start with general behaviour and add the real-time and real-space aspects at a later stage

Components and abstraction level

Why is structure necessary?

- interaction between components
- communication mechanisms: asynchronous (place fusion) or synchronous (transition fusion)

Advantages

- reusability: try out several models for a single component
- use of modular analysis techniques

Abstraction

- apply a refinement process
- add details step by step until the desired abstraction level is reached
- start with general behaviour and add the real-time and real-space aspects at a later stage

Dynamic actors

How to handle dynamicity?

create new vehicles getting in and discard those getting out

not suitable

- numbers associated with vehicles ⇒ state space explosion (even if there is a maximal id)
- identities are not important as long as the vehicles can be distinguished

maximal number of vehicles in the system (due to physical reasons)

⇒ reuse identity of exiting vehicles

Dynamic actors

How to handle dynamicity?

create new vehicles getting in and discard those getting out

not suitable

- numbers associated with vehicles ⇒ state space explosion (even if there is a maximal id)
- identities are not important as long as the vehicles can be distinguished

maximal number of vehicles in the system (due to physical reasons)

⇒ reuse identity of exiting vehicles

Adding real-time and real-space aspects

Complex functions

1. discretisation of complex functions
2. use timed or hybrid formalisms

Fair execution of components

- use of a timeline as in timed Petri nets
- state space construction with branching criteria discarding unsuitable sequences

Adding real-time and real-space aspects

Complex functions

1. discretisation of complex functions
2. use timed or hybrid formalisms

Fair execution of components

- use of a timeline as in timed Petri nets
- state space construction with branching criteria discarding unsuitable sequences

More advanced analysis techniques

Analysis techniques inadequate for systems where types have many values (due to discretisation)

⇒ more elaborate techniques:

- symbolic/symbolic approaches combine:
  - symbolic reachability graphs
  - symbolic encoding of states
  - sharing features
  - hierarchical structuring
- distributed model-checkers running on clusters of machines

More advanced analysis techniques

Analysis techniques inadequate for systems where types have many values (due to discretisation)

$\Rightarrow$ more elaborate techniques:

- symbolic/symbolic approaches combine:
  - symbolic reachability graphs
  - symbolic encoding of states
  - sharing features
  - hierarchical structuring
- distributed model-checkers running on clusters of machines

Similarities between ITS and autonomous robots

- robots evolve in an environment which may have unexpected behaviour
- information obtained from sensors gives an abstract vision of the environment
- real-time and real-space features

Analogies between ITS and autonomous robots

- robots ↔ vehicles
- interacting agents ↔ infrastructure

Similarities between ITS and autonomous robots

- robots evolve in an environment which may have unexpected behaviour
- information obtained from sensors gives an abstract vision of the environment
- real-time and real-space features

Analogies between ITS and autonomous robots

- robots $\leftrightarrow$ vehicles
- interacting agents $\leftrightarrow$ infrastructure