A self-certifiable architecture for critical systems powered by probabilistic logic artificial intelligence

Jacques Robin, Raul Mazó, Daniel Diaz
Université Paris 1 Panthéon-Sorbonne
Henrique Madeira, Raul Barbosa
Universidade de Coimbra
Salvador Abreu
Universidade de Évora

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

UNIVERSIDADE Ð COIMBRA

UNIVERSIDADE DE ÉVORA

AI4EU

Outline

1. The problem: certification of online learning critical systems
2. Versatile rules for self-certifiable AI
3. A self-certifiable autonomic architecture for critical AI
4. Limitations and future work

Certification of AI-powered critical systems

Current critical systems engineering

- **Two human** dependability **experts** negotiate: an engineer and an auditor
- On the basis of **natural language** documentation
- Presenting **evidence** that:
  - The critical system's probability of **failure is below threshold** required by the industry's dependability standard
  - The process to engineer the system **correctly instantiates** the abstract process prescribed by the standard
- Human industry-specific experts:
  - Write detailed, **explicit requirements**
  - Specify full software control flow
  - Write **systematic tests** against this flow
- Once deployed, system behaviour is assumed fixed
- Certification can hence occur only once, prior to deployment

Engineering AI-powered critical systems

- AI decomposed into 4 components:
  - Industry-independent **inference engine**
  - System-specific declarative **knowledge base**
  - Industry-independent **machine learning algorithms**
  - System-specfic **data sets** and mining process from which to learn declarative knowledge
- Control flow emerges through interaction of those 4 components
- Requirements may be specified only *extensionally* as training datasets
- Testing I/O pairs of AI component may not be manually specifiable
- Dependability analysis needs rethinking
- **Online learning** supports implementing smartest critical systems that autonomically self-adapt to context changes, some unforeseen at design time
- Each online learned knowledge sentence pushed to operational critical system triggers need for re-certfiication
- **Certification automation crucial to contain re-certification cost**

Versatile rule language for self-certifiable AI

- Probabilistic logic constraint solving rules *(CHRiSM, Sneyers et al. 09,10)*
- Engine apply rules to transform initial constraint store containing *Constraint Satisfaction Problem (CSP)* into final constraint store containing *CSP Solution (CSPS)* or failure (when CSP is overconstrained)
- CSP and CSPS: $\bigwedge_m c_m(L_n)$ with L_n logical variables, c_m relations/constraints
- When CSP exactly constrained, CSPS: $\bigwedge_n L_n = k_n$ with k_n constants
- Constraint *simplification/rewrite* rule:

$$p(L_u, R_v) :: (\bigwedge_i g_i(L_w) \rightarrow \left(\bigwedge_j h_j(L_x\right) \leftrightarrow \bigvee_k (q_k(L_k{}^y, R_k{}^z) :: \bigwedge_l b_k{}^l(L_k{}^t)))) \text{ with}$$

  - $p(L_u, R_v)$, $q_k(L_k{}^y, R_k{}^z)$ probability expressions in [0,1]
  - R_v, $R_k{}^z$ random variables, L_u, L_w, L_x, L_y, $L_k{}^y$, $L_k{}^t$ logical variables
- Constraint *propagation/production* rule:

$$p(L_u, R_v) :: (\bigwedge_i g_i(L_w) \rightarrow \left(\bigwedge_j h_j(L_x\right) \rightarrow \bigvee_k (q_k(L_k{}^y,\ R_k{}^z) :: \bigwedge_l b_k{}^l(L_k{}^t))))$$

- Rule triggers with probability $p(L_u, R_v)$ when:
  - Rule head $\bigwedge_j h_j(L_x) \subseteq$ store $\bigwedge_m c_m(L_n)$ (modulo variable pattern matching), and
  - Rule guard $\bigwedge_i g_i(L_w)$ |= store $\bigwedge_m c_m(L_n)$ (modulo variable pattern matching)
- Triggered:
  - Simplification rule *substitutes* store subset matchng rule head with rule body $\bigwedge_l b_k{}^l(L_k{}^t)$ (modulo variable pattern matching) with probability $q_k(L_k{}^y,\ R_k{}^z)$
  - Propagation rule *adds* rule body $\bigwedge_l b_k{}^l(L_k{}^t)$ (modulo variable pattern matching) to store with probability $q_k(L_k{}^y,\ R_k{}^z)$, *keeping* store subset matching rule head in the store

Versatile rule engine for self-certifiable AI

- CHRISM engine queries:
  - **solve(S_i,S_f)** to probabilistically search solution S_f to CSP S_i
  - **prob($S_i \Leftrightarrow S_f$, P)** to compute probability P of S_f being solution to CSP S_i
  - **learn(E,R,D)** to learn from set E of example pairs (*CSP,CSPS)* the probability distribution D of the random variables R in the probability expressions of a CHRISM rule based that transforms initial store CSP into final store CSPS

- Example CHRISM rule base encoding of classical alarm toy Bayesian net:
  - $go \Rightarrow P_b::burglary(yes) \vee (1\text{-}P_b)::burglary(no)$
  - $go \Rightarrow P_e::earthquake(yes) \vee (1\text{-}P_e)::earthquake(no)$
  - $burglary(B) \wedge earthquake(E) \Rightarrow P_a(B,E)::alarm(yes) \vee (1\text{-}P_a(B,E))::alarm(no)$
  - $P_j(A)::(alarm(A) \Rightarrow johncalls)$
  - $P_m(A)::(alarm(A) \Rightarrow marycalls)$

- Query:
  - $prob(\{go\} \Leftrightarrow \{go, burglary(no), earthquake(yes), alarm(yes), marycalls\}, P)$
  - Instantiates *P* with $(1\text{-}P_b)*P_e*P_a(no,yes)*P_m(yes)$

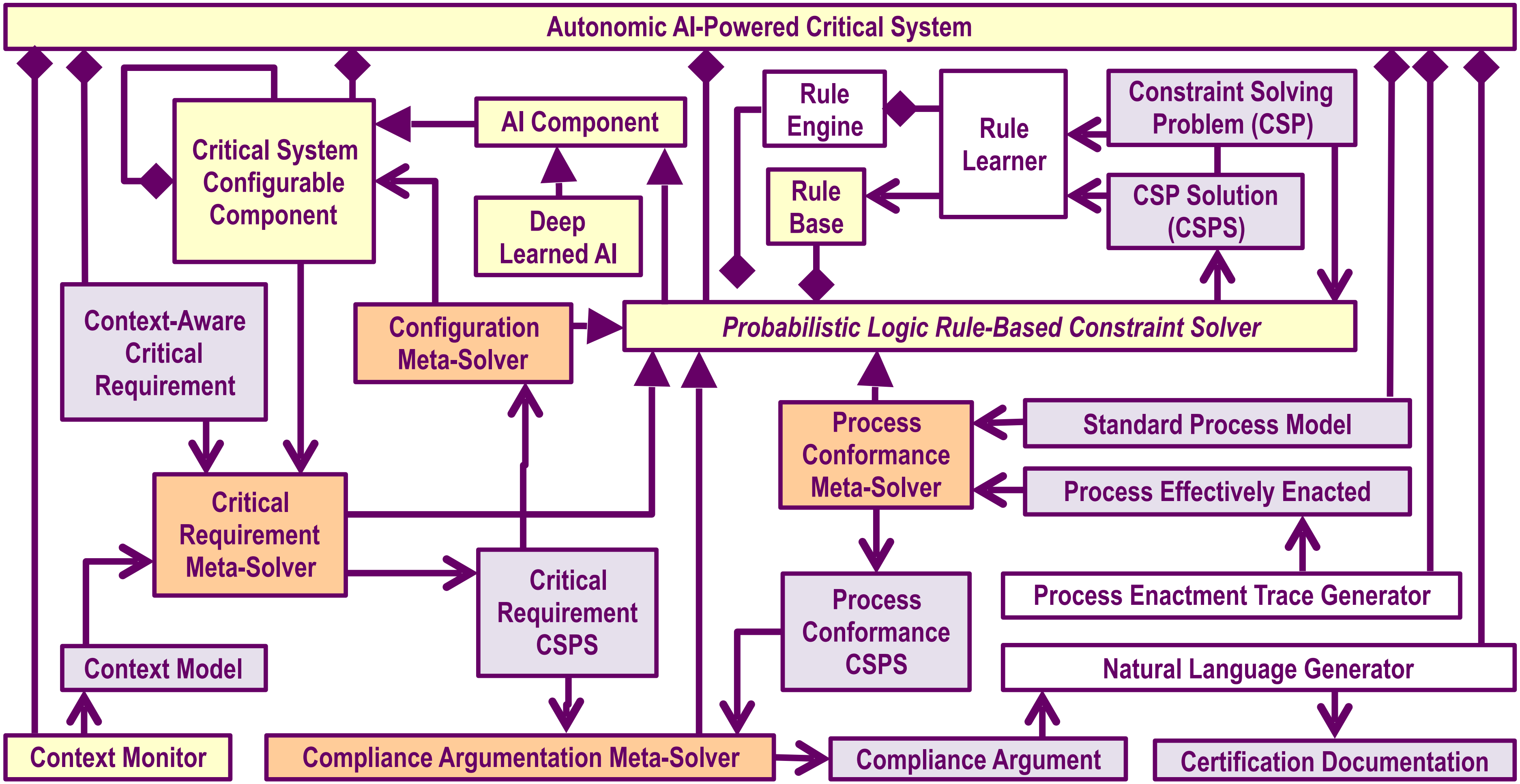
Versatile rules for self-certifiable AI

- CHRISM rules generalize:
  - CHR$^{\vee}$ rules with probabilities
  - CHR$^{\vee}$ rules themselves generalize:
    - CHR rules with disjunctive bodies (CHR$^{\vee}$ engine adds search to CHR engine) [Frühwirth 09]
    - *Constraint Logic Programming (CLP)* rules with guards which can serve as connecting interfaces in assembly of encapsulated rule-based components [Fages et al. 09]
    - CHR rules themselves generalize term rewriting and production/business rules [Frühwirth 09]
    - Frame logics and description logics (ontologies) [Almeida, Robin 09], [Frühwirth 09]
  - Relational Bayes nets with guards and local logical semantic reading
    - Relational Bayes nets themselves generalize propositional Bayes nets with universally quantified logical variables

- CHRISM engine generalizes:
  - CHR$^{\vee}$ engine with probabilistic reasoning and machine learning

- CHR$^{\vee}$ and CLP engines successfully used for:
  - Optimization [Bistarelli et al. 04], constraint solving [Frühwirth 09]
  - Deduction [Duck 12], abduction [Christiansen 08]
  - Natural Language Processing [Christiansen 05]
  - Belief update [Thieslcher 06], belief revision [Jin, Thielscher 07]
  - Default reasoning [Almeida et al. 08], argumentative reasoning [Sneyers et al. 13]
  - Ontological reasoning [Almeida, Robin 09], [Frühwirth 09]

Autonomic architecture for self-certifiable AI



Limitations and future research agenda

- Not yet evaluated on case study
- Current CHRISM engine:
  - Lacks rule structure learning
  - Lacks interface with deep learned AI components
  - Implemented only in Prolog

- Evaluation on:
  - AI4EU railway control system cybersecurity case study
  - AI4EU radiology assistant case study
  - AI4EU optimised smart factory
- Basis for CHRISM engine extension feasibility:
  - Structure learning of ProbLog and CP-Logic rules [Riguzzi 18]
  - ProbLog interface with deep learning [Manhave et al 18]
  - CHR to VHDL compiler [Triossi et al 12]
  - CHR compilers to Haskell, JavaScript, Java, C