
Global Caching for Coalgebraic Description Logics

Dirk Pattinson, Imperial College London

(Joint work with Rajeev Gorè, Clemens Kupke and Lutz Schröder)

Edinburgh, July 2010

Back in Tudor England ...



Henry VIII



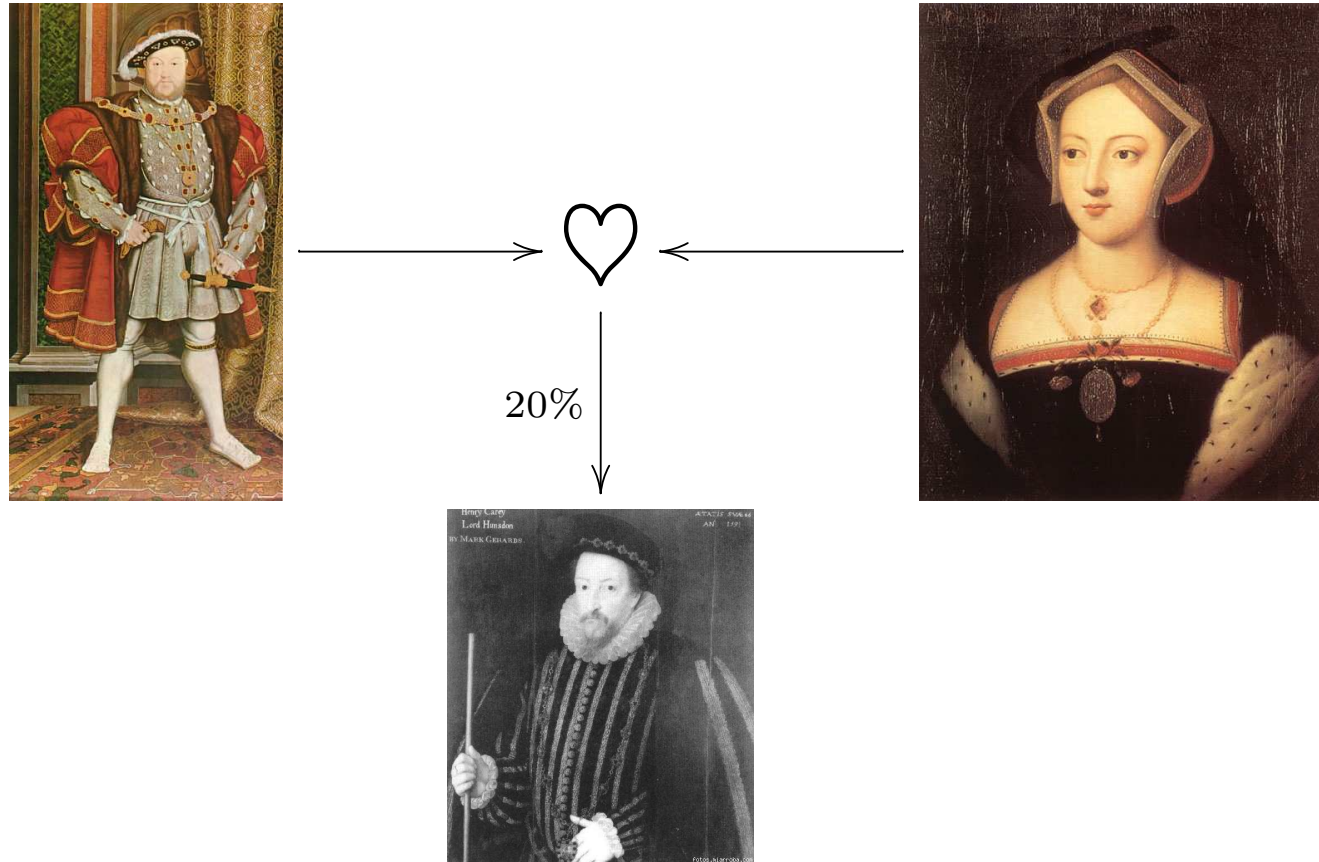
Henry Carey



Mary Boleyn

“There has been speculation that Mary’s two children, Catherine and Henry, were fathered by Henry, but this has never been proven”

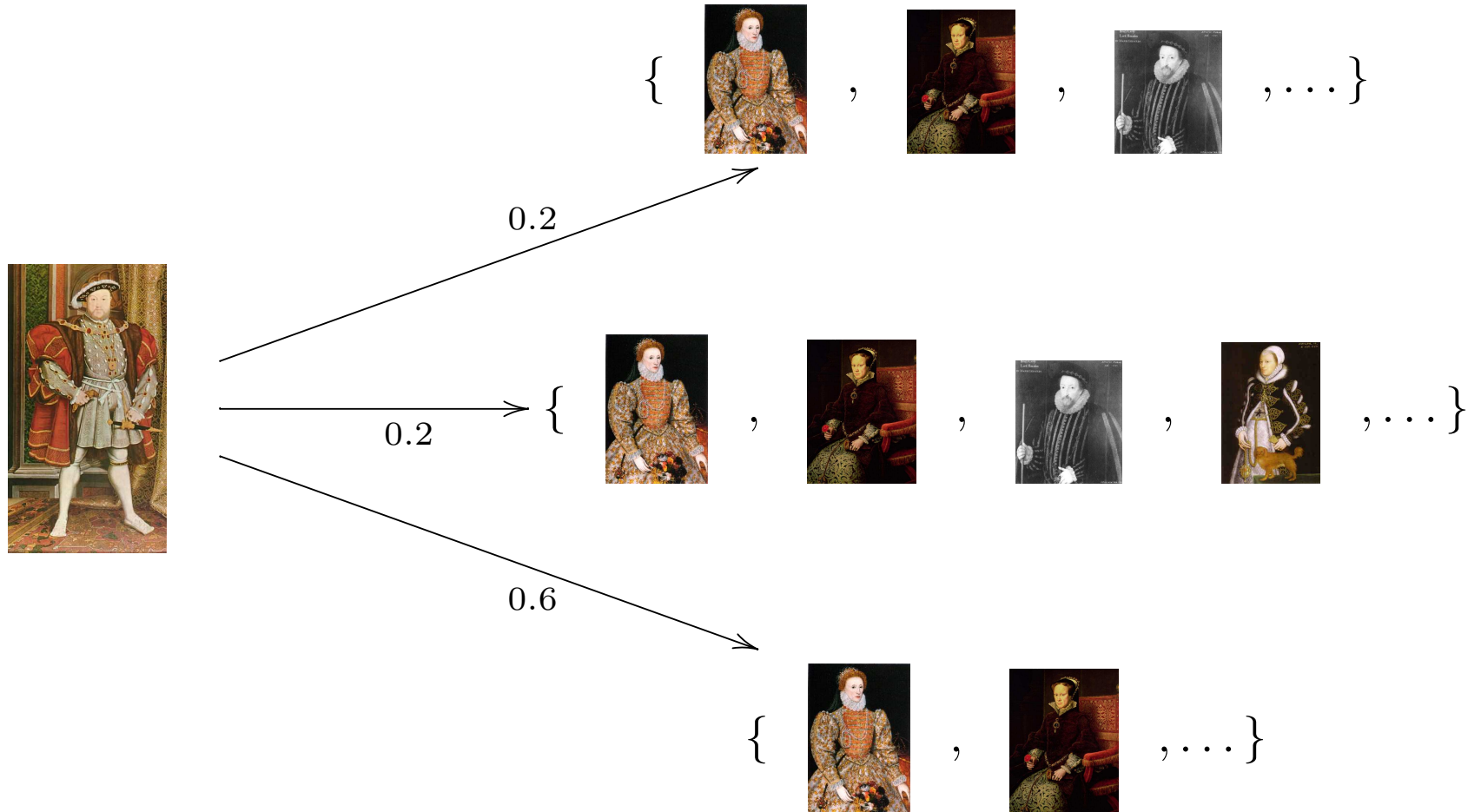
What do we know?



- Mary Boleyn was Henry's Mistress time between 1520 and 1526 (approx.)
- suppose that there's a *20 % chance* that Henry Carey is a royal bastard

What's a good model?

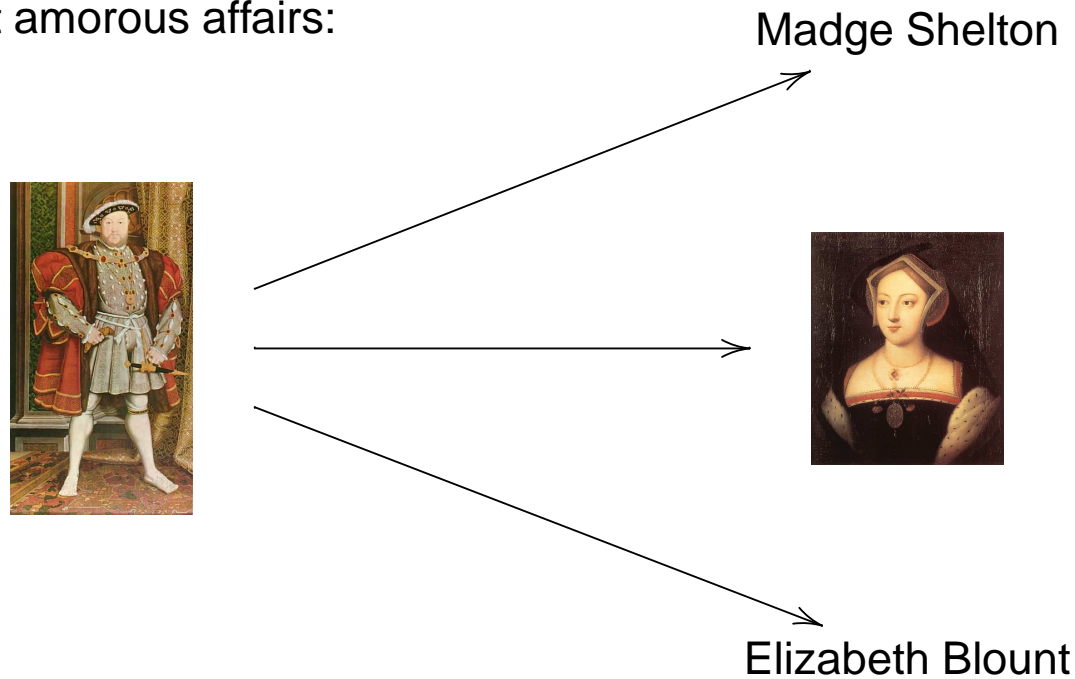
Quantitative Uncertainty about offspring:



Models are *probability distributions* over *sets of successors*: $W \rightarrow \mathcal{D}(\mathcal{P}(W))$

What's a good model?

Certainty about amorous affairs:



Models are *relations*: $W \rightarrow \mathcal{P}(W)$

Combinations of both facets:

$$W \rightarrow \underbrace{\mathcal{D}(\mathcal{P}(W))}_{\text{offspring}} \times \underbrace{\mathcal{P}(W)}_{\text{affairs}}$$

Logical Language

Syntax based on

- a set P of propositional variables (concepts) like king
- a set N of nominals (individual variables) like henry
- a set Λ of modal operators (like “had an affair with”)

Formulas

$\mathcal{H}(\Lambda) \ni A, B ::= x \mid A \wedge B \mid A \vee B \mid \heartsuit(A_1, \dots, A_n) \mid \heartsuit^{\bar{}}(A_1, \dots, A_n) \mid @_i A$

where $x \in P \cup \bar{P} \cup N \cup \bar{N}$, $i \in \mathbb{N}$ and $\heartsuit \in \Lambda$ is n -ary.

Interpretation.

- nominals denote singletons, $@_i$ moves evaluation context to point i
- modalities “scan successors” for relevant properties item $M, w \models A$ for some $w \in W$ — A is satisfiable in M

How do we interpret this?

Tudor Example. $\Lambda = \{\diamond\} \cup \{L_p \mid p \in \mathbb{Q} \cap [0, 1]\}$

Models are triples $M = (W, \sigma, \pi)$ where

$$\sigma : W \rightarrow TW = \mathcal{D}(\mathcal{P}(W)) \times \mathcal{P}(W) \quad \text{and} \quad \pi : P \cup N \rightarrow \mathcal{P}(W)$$

are such that $\pi(n)$ is a singleton for all $n \in N$.

Modalities (where $\llbracket A \rrbracket_M = \{w \in W \mid M, w \models A\}$ is the truth-set of A)

$$M, w \models \diamond A \iff \sigma(w) \in \{(\mu, S) \in TW \mid S \cap \llbracket A \rrbracket_M \neq \emptyset\}$$

$$M, w \models L_p A \iff \sigma(w) \in \{(\mu, S) \in TW \mid \mu(\{S' \subseteq W \mid S' \cap \llbracket A \rrbracket_M \neq \emptyset\}) \geq p\}$$

Satisfaction Operators and Variables (where $x \in P \cup N$)

$$M, w \models @_i A \iff M, \pi(i) \models A \quad M, w \models x \iff w \in \pi(x)$$

Enter Coalgebra ...

Models in the Tudor example: $M = (W, \sigma, \pi)$ where

$$\sigma : W \rightarrow TW = \mathcal{P}(\mathcal{D}(W)) \times \mathcal{P}(W)$$

Modalities in the Tudor Example:

$$M, w \models \diamond A \iff \sigma(w) \in \{(\mu, S) \in TW \mid S \cap \llbracket A \rrbracket_M \neq \emptyset\}$$

$$M, w \models L_p A \iff \sigma(w) \in \{(\mu, S) \in TW \mid \mu(\{S' \subseteq W \mid S' \cap \llbracket A \rrbracket_M \neq \emptyset\}) \geq p\}$$

Coalgebraic Models. $M = (W, \sigma, \pi)$ where $\sigma : W \rightarrow TW$

Coalgebraic Modalities.

$$M, w \models \heartsuit(A) \iff \sigma(w) \in \llbracket \heartsuit \rrbracket_W(\llbracket A \rrbracket_M)$$

and $\llbracket \heartsuit \rrbracket$ is a *predicate lifting*, i.e. a natural family of mappings of type

$$\llbracket \heartsuit \rrbracket_X : \mathcal{P}(X) \rightarrow \mathcal{P}(TX)$$

Coalgebraic Setup

Given.

- a collection Λ of modal operators
- a functor $T : \text{Set} \rightarrow \text{Set}$ inducing T -models $(C, \gamma : C \rightarrow TC)$
- an interpretation $\llbracket \heartsuit \rrbracket : \mathcal{P} \rightarrow \mathcal{P} \circ T$ of every $\heartsuit \in \Lambda$

Problem. Decide whether $A \in \mathcal{H}(\Lambda)$ is satisfiable in $\text{Mod}(\Xi)$, for $\Xi \subseteq \mathcal{H}(\Lambda)$.

Moreover, what's the complexity and what's a feasible algorithm?

Examples.

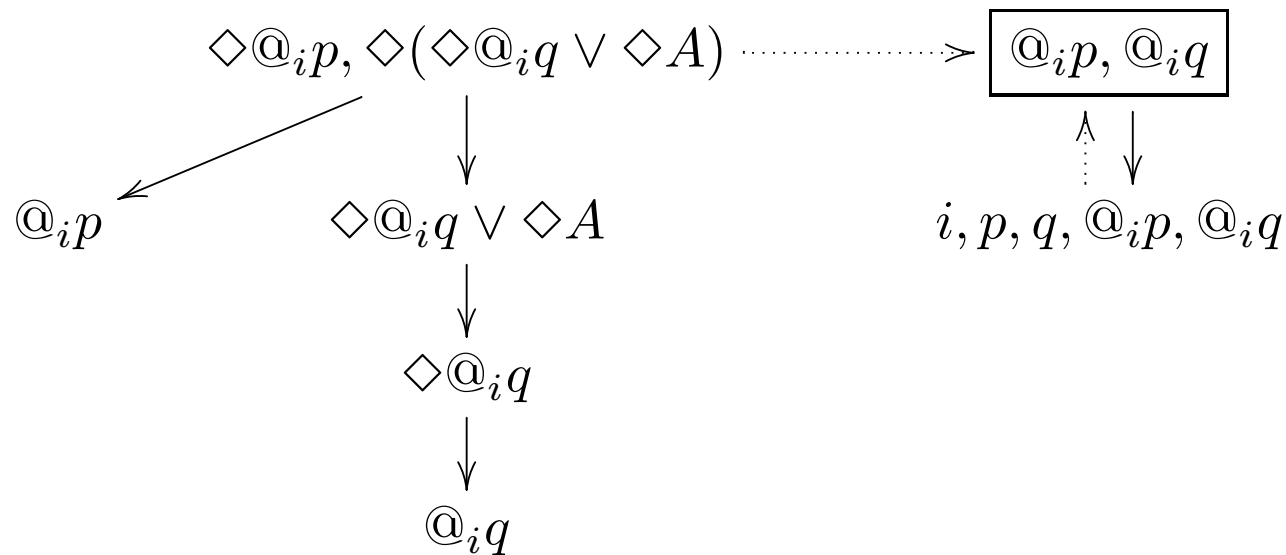
- the Tudors (as we've seen before)
- and many more by varying T and Λ (graded, multi-agent, conditional etc.)

Coalgebraic Tableaux in Action

Basic Idea. Satisfiability as a game played by \exists (claiming sat) and \forall

- every rule application (challenge of \forall) must have a satisfiable conclusion (\exists)
- rule application may uncover @-prefixed formulas that need to be propagated
- propagated @-formulas can be challenged by \forall

Example 1. \exists shows satisfiability

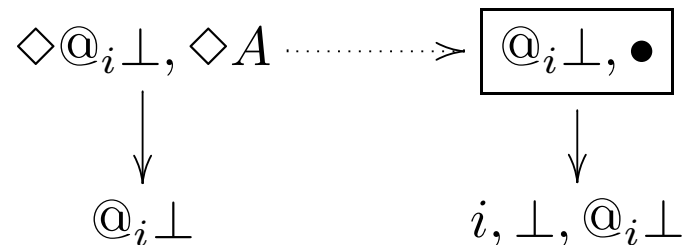


Coalgebraic Tableaux in Action

Basic Idea. Satisfiability as a game played by \exists (claiming sat) and \forall

- every rule application (challenge of \forall) must have a satisfiable conclusion (\exists)
- rule application may uncover @-prefixed formulas that need to be propagated
- propagated @-formulas can be challenged by \forall

Example 2. \forall shows unsatisfiability



Note. In both examples, wins were announced without fully unfolding the tableau.

Technicalities: Caching Graphs

Basic Entities.

- *Sequents* are finite sets of formulas
- *@-constraints* are finite sets of @-prefixed formulas possibly containing •

[• signifies incomplete information]

Expansion Rules

- *for sequents*: propositional logic + modal rules (logic specific, but known)
- *for @-constraints*:

$$\frac{\Upsilon}{i, \Upsilon / @_i, \Upsilon \setminus \{\bullet\}}$$

where $i \in N(\Upsilon)$ is a nominal occurring in Υ and $\Upsilon / @_i = \{A \mid @_i A \in \Upsilon\}$.

[Υ is needed as further constraints may be uncovered by modal unfolding]

Expansion and Propagation in a Nutshell

Sequent Expansion \rightarrow_S

- select unexpanded sequent (X) and create/reuse nodes for all applicable rules
- create associated @-constraint containing just •
- mark sequent as unknown (U)

Constraint Expansion \rightarrow_C .

- select unexpanded constraint (T) and create/reuse nodes for all applicable rules
- mark constraint as done (D)

Constraint Propagation \rightarrow_P

- percolate constraints through the graph using greatest-fixpoint construction

Position Update \rightarrow_U .

- Update winning positions for \forall *inductively*: add sequents that are unsat
- Update winning positions for \exists dually (using *coinduction*)

Properties of Ensuing Algorithm

Correctness. Given that the modal rules are sound and complete, sequents marked 'satisfiable' are indeed satisfiable (dually for unsat).

Proof. If a sequent is marked 'sat', player \exists has a winning strategy in a game played on sequents, which induces a satisfying model.

Termination. If modal rules are decidable, every execution terminates and all sequents will be marked as either sat or unsat.

Proof. Determinacy of the associated two-player game.

Complexity. If modal rules are EXPTIME decidable, every execution of the algorithm will terminate in EXPTIME.

Proof. The number of sequents that appear in the game is exponential in the size of the root sequent.

Non-Determinism leaves room for heuristics.

Conclusions

Complexity. For coalgebraic logics, satisfiability is EXPTIME-decidable over a (finite) set of global assumptions

- this has been known before, but we have a new proof

Practicability. The decision procedure is purely syntax driven

- amenable to ‘usual’ optimisations for tableau-bases algorithms
- room for heuristics

Novelty. Even for (description) logics with relational semantics, our algorithm appears to be new?

Still Missing (but not for long): implementation and experiments.