AN ONTOLOGICAL APPROACH TO A SYSTEM OF REQUIREMENTS PATTERNS

NATALIA GARANINA

A.P. ERSHOV INSTITUTE OF INFORMATICS SYSTEMS, NOVOSIBIRSK, RUSSIA

INTRODUCTION

- The project *Methods for generation of formal models and requirements from technical documentation presented in a natural language and their verification*.
  - Improving the quality of software systems using formal methods.
  - Extracting formal models and properties of distributed software systems
    - from texts of technical documentation
  - Verification the properties in the models.
- Development
  - an ontology of distributed software systems
  - ontologies of requirements
    - requirements
    - information about the methods and complexity of their verification.
  - methods of information extraction to populate these ontologies.

INTRODUCTION

- Syntactical classification [Manna, Pnueli 91]
  - liveliness, safety, etc
- Qualitative requirements [Dwyer, Avrunin, Corbett 99]
  - absence, existence, universality, precedence, responce
- Real-time requirements [Konrad, Cheng 05]
- Probabilistic requirements [Grunske 08].
- Composite event templates [Mondragon, et. al. 04] [Salamah, et. al. 12]
- Quantitative features of events' appearance [Bianculli, et. al. 12]
  - patterns for data [Halle, 09]

- Patterns expressible in LTL and its real-time and probabilistic extensions

INTRODUCTION

- Branching time requirements [Post, Menzel, Podelski 11]
- Combined patterns [Autili , et. al. 15]
  - classical, probabilistic, and real-time patterns
  - their description in restricted English
- Pattern ontology [Yu, Manh, Han 06]
  - qualitative requirements
  - composite events

INTRODUCTION

- Two ontologies:
  - a handbook of patterns
    - known pattern systems + new patterns
  - an ontology of requirements
    - unique for each individual set of technical documentation
  - presented in OWL.

INTRODUCTION

The ontology-handbook of patterns

- Knowledge organization system of specification patterns.
  - informal description of the patterns
  - formal descriptions in logics
    - graphical representations (GIL, UML),
  - examples of usage,
  - parameters,
  - verification complexity,
  - suitable verification tools,
  - application areas,
  - information on algorithms of realizability, etc.
- This ontology is populated manually
- Ambiguity of a natural language
  - semantics of different representations may differ.

INTRODUCTION

The ontology of requirements

- Classes of requirements with corresponding parameters
  - Attribute values are extracted from the texts.
  - The names of the classes from the handbook of the patterns.
    - an extracted requirement corresponds to its pattern in the handbook
      - correctness of formulation
      - model checking.
- Population from technical documentation texts.

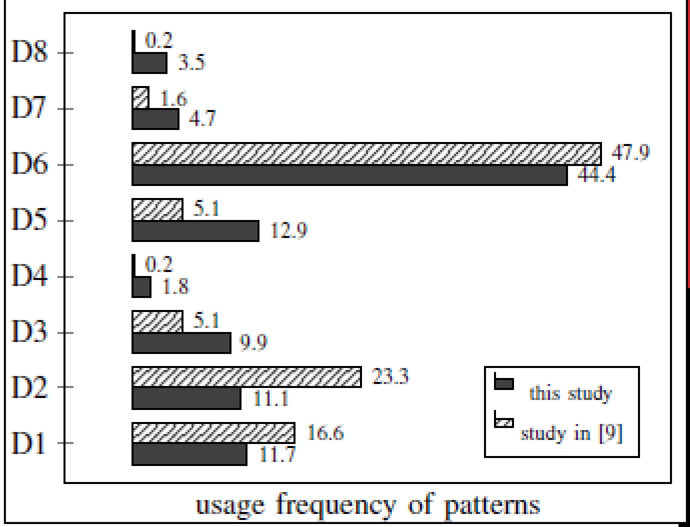
INTRODUCTION

- Specification of requirements
  - a Boolean combination of the following pattern types:
    - qualitative,
    - real-time,
    - branching time,
    - complex events,
    - quantitative features of events,
    - simple statements about the data,
    - optimality,
    - undesirable behavior of an environment.

SYSTEMS OF PATTERN CLASSIFICATION

(1) Qualitative Patterns

- Occurrence patterns
  - appearance of a system event P during system execution.
- Absence: $\mathbf{G}\neg P$
  - it is never the case that P holds
- Universality: $\mathbf{G}P$
  - it is always the case that P holds
- Existence: $\mathbf{F}P$
  - P eventually holds
- Bounded existence: $P\ \mathbf{W}\ (P\ \mathbf{W}\ (\neg P\ \mathbf{W}\ (P\ \mathbf{W}\ \mathbf{G}\neg P)))$
  - it is always the case that the transitions to state P occur at most 2 times



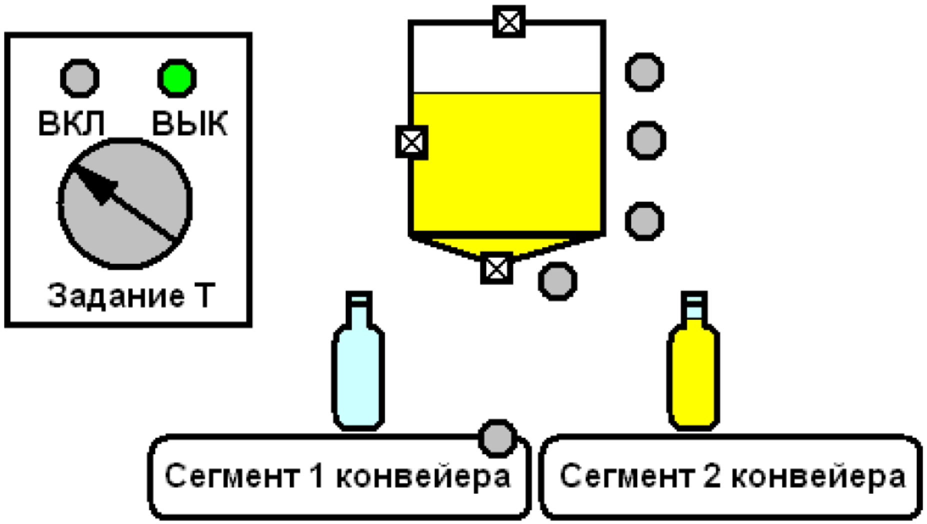
SYSTEMS OF PATTERN CLASSIFICATION

(1) Qualitative Patterns

- Order patterns
  - the relative sequence of appearance of events P and S during execution.
- Precedence: $\mathbf{F}P \rightarrow \neg P\mathbf{U}(S \wedge \neg P)$
  - it is always the case that if P holds, then S previously held
- Response: $\mathbf{G}(P \rightarrow \mathbf{F}S)$
  - it is always the case that if P holds, then S eventually holds
- Response and precedence chains: $\mathbf{G}(P \rightarrow (S \wedge \mathbf{XF}T))$
  - a generalization of the corresponding patterns
    - relationships between sequences of individual states/events.
- Constrained chain
  - restricts user specified events from occurring between pairs of states/events in the chain sequences.

ПРИМЕР

Автоматизированная линия розлива бутылок



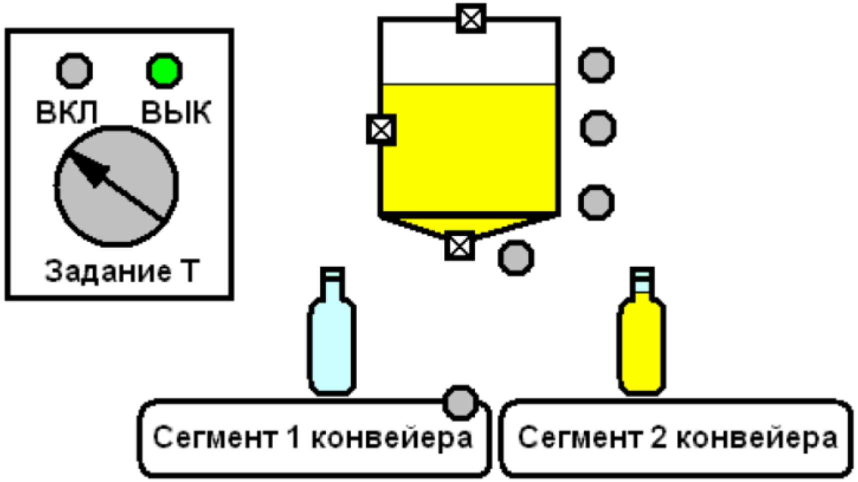
(1, 2) - сегменты конвейера, (3) - кнопка управления, (6) - датчик положения бутылки, (7) - клапан разлива в бутылку

1. Включается система (3), включаются конвейеры (1, 2).

2. Пустая бутылка подъезжает к клапану разлива, срабатывает датчик положения бутылки (6), конвейер 1 останавливается, включается клапан разлива (7), бутылка наполняется.

EXAMPLE

- Existence: **F** P_0
  - P_0 eventually holds
- P_0 – жидкость нагревается



- Response: **G**(P_1 → **F** P_2)
  - it is always the case that if P_1 holds, then P_2 eventually holds
- P_1 – появление бутылки
- P_2 – наполнение бутылки

SYSTEMS OF PATTERN CLASSIFICATION

(1) Qualitative Patterns.

- Additional constraints on appearance of events P and S in the order patterns.
  - Non-overlap: $\mathbf{G}((P \rightarrow \neg S) \wedge (S \rightarrow \neg P))$
    - P and S never hold simultaneously
    - P_1 non-overlap P_2
  - Right-overlap: $\mathbf{G}((P \rightarrow \neg S) \wedge \mathbf{X}(P \rightarrow \neg S \ \mathbf{U} \ (S \wedge \mathbf{X}(S \ \mathbf{U} \ \neg P)))$
    - it is always the case that P holds before S and there is a finite state sequence in which P and S hold simultaneously.
  - Left-overlap: $\mathbf{G}((S \rightarrow \neg P) \wedge \mathbf{X}(S \rightarrow \neg P \ \mathbf{U} \ (P \wedge \mathbf{X}(P \ \mathbf{U} \ \neg S)))$
    - it is always the case that S holds before P and there is a finite state sequence in which P and S hold simultaneously

- P_1 – появление бутылки
- P_2 – наполнение бутылки

SYSTEMS OF PATTERN CLASSIFICATION

(2) Scopes.

- Patterns – what, scopes -- when.
- Globally: **G**P
  - a pattern holds throughout the program execution.
- Before R: **F**R → P**U**R
  - a pattern holds during program execution before R first occurs.
- After R: **G**(R → **G**P)
  - a pattern holds during program execution after R first occurs.

SYSTEMS OF PATTERN CLASSIFICATION

(2) Scopes.

- Between Q and R:
  - a pattern holds during execution between the first occurrence of Q and the next occurrence of R.
  - Q_1 – система включена
  - Q_2 – система выключена
  - $\mathbf{G}((Q_1 \wedge \neg Q_2 \wedge \mathbf{F}\, Q_2) \rightarrow (P_1 \rightarrow (\neg Q_2\, \mathbf{U}\, (P_2 \wedge \neg\, Q_2))\, \mathbf{U}\, Q_2))$

- P_1 – появление бутылки
- P_2 – наполнение бутылки

SYSTEMS OF PATTERN CLASSIFICATION

(2) Scopes.

- Between Q and R: $\mathbf{G}((Q \wedge \neg R \wedge \mathbf{F}\, R) \rightarrow P\ \mathbf{U}\ R)$
  - a pattern holds during execution between the first occurrence of Q and the next occurrence of R.
- After Q until R: $\mathbf{G}((Q \wedge \neg R) \rightarrow P\ \mathbf{W}\ R)$
  - a pattern holds during execution between the first occurrence of Q and until the next occurrence of R or the end of the program execution.
- Start (initial phase): $\neg R \rightarrow P\ \mathbf{W}\ R$
  - a pattern holds until event R marks the end of the initial phase.
- Final (final phase): $(\mathbf{FG}\, R \wedge \mathbf{F}\, P) \rightarrow \neg\, P\ \mathbf{U}\ R\ )$
  - a pattern holds after event R marks the beginning of the final phase.
- Regular (repeating phase) = After-until scope.

SYSTEMS OF PATTERN CLASSIFICATION

(3) Branching time.

- Possible Existence: **AG**(P→ **EF** S)
  - if P holds then there is at least one execution sequence such that S eventually holds.
- Possible Universality: **AG**(P→ **EG** S)
  - if P holds then there is at least one execution sequence such that S forever holds.
- Possible Precedence: **EF**(P→ **AF** S)
  - if S holds then there is at least one execution sequence such that P holds before.
- Possible Response: **AF**(P→ **EF** S)
  - if P holds then there is at least one execution sequence such that S holds.

- P_0 – жидкость нагревается

SYSTEMS OF PATTERN CLASSIFICATION

(4) Real Time.

Duration

- Point: $\mathbf{FG}_{=k}P$
  - it's always the case that once P becomes satisfied, it holds exactly k time units.
- Minimum: $\mathbf{FG}_{\geq k}P$
  - it's always the case that once P becomes satisfied, it holds at least k time units.
    - $\mathbf{FG}_{\geq k}\,P_0$
- Maximum: $\mathbf{FG}_{\leq k}P$
  - it's always the case that once P becomes satisfied, it holds less than k time units.
- Interval: $\mathbf{FG}_{km}P$
  - it is always the case that once P becomes satisfied, it holds at least k time units and less than m time units.

- P_0 – жидкость нагревается

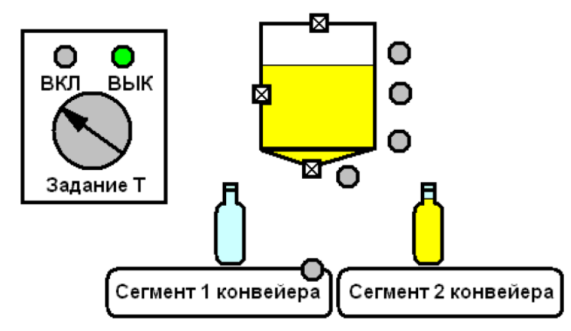
SYSTEMS OF PATTERN CLASSIFICATION

(4) Real Time.

- Periodic: $\mathbf{GF}_{\leq \mathbf{k}}P$
  - it is always the case that P holds at least every k time units.
    - $\mathbf{GF}_{\leq \mathbf{k}}\, P_0$
- Absolute time
  - 26.06.2017

SYSTEMS OF PATTERN CLASSIFICATION

- P_2 – наполнение бутылки
  - R_1 – конвейер 1 движется
  - R_2 – датчик положения бутылки сработал
  - R_3 – конвейер 1 стоит
  - R_4 – клапан розлива открыт



(5) Complex events.

- Complex events are compositions of elementary events from set E.
  - concurrency, non-determinism, and interval of events at some state of system execution.
- One(E), strict mode: $e_1 \vee e_2 \vee \ldots \vee e_n$
  - at least one of the propositions in the set E holds.
- One(E), free mode: $(\neg e_1 \wedge \mathbf{F}e_1) \vee (\neg e_2 \wedge \mathbf{F}e_2) \vee \ldots \vee (\neg e_n \wedge \mathbf{F}e_n)$
  - at least one of the propositions in the set E becomes true.

SYSTEMS OF PATTERN CLASSIFICATION

(5) Complex events.

- Complex events are compositions of elementary events from set E.
  - concurrency, non-determinism…
- One(E), strict mode: $e_1 \vee e_2 \vee \ldots \vee e_n$
  - at least one of the propositions in the set E holds.
  - $P_2 = R_1 \vee R_2 \vee R_3 \vee R_4$
- One(E), free mode: $(\neg e_1 \wedge \mathbf{F} e_1) \vee (\neg e_2 \wedge \mathbf{F} e_2) \vee \ldots \vee (\neg e_n \wedge \mathbf{F} e_n)$
  - at least one of the propositions in the set E becomes true.
  - $P_2 = (\neg R_1 \wedge \mathbf{F} R_1) \vee (\neg R_2 \wedge \mathbf{F} R_2) \vee (\neg R_3 \wedge \mathbf{F} R_3) \vee (\neg R_4 \wedge \mathbf{F} R_4)$

- R_1 – конвейер 1 движется
- R_2 – датчик положения бутылки сработал
- R_3 – конвейер 1 стоит
- R_4 – клапан розлива открыт

SYSTEMS OF PATTERN CLASSIFICATION

(5) Complex events.

- Parallel(E), strict mode: $E = e_1 \wedge e_2 \wedge \ldots \wedge e_n$
  - all propositions in the set E hold.
  - $P_2 = R_1 \wedge R_2 \wedge R_3 \wedge R_4$
- Parallel(E), free mode: $(\neg e_1 \wedge \neg e_2 \wedge \ldots \wedge \neg e_n) \wedge (\neg E \mathbf{U} E)$
  - all propositions in the set E become true.
  - $P_2 = (\neg R_1 \wedge \neg R_2 \wedge \neg R_3 \wedge \neg R_4) \wedge (\neg E \ U \ R_1 \wedge R_2 \wedge R_3 \wedge R_4)$

- R_1 – конвейер 1 движется
- R_2 – датчик положения бутылки сработал
- R_3 – конвейер 1 стоит
- R_4 – клапан розлива открыт

- R_1 – конвейер 1 движется
- R_2 – датчик положения бутылки сработал
- R_3 – конвейер 1 стоит
- R_4 – клапан розлива открыт

SYSTEMS OF PATTERN CLASSIFICATION

(5) Complex events.

- Serial(E), strict mode: $e_1 \wedge \mathbf{X}(e_2 \wedge \mathbf{X}(e_3 \ldots \wedge \mathbf{X} e_n)\ldots)$
    - Each proposition in the sequence E is asserted to hold in a specified order, one at each successive state.
    - $P_2 = R_1 \wedge \mathbf{X}(R_2 \wedge \mathbf{X}(R_3 \wedge \mathbf{X} R_4))$
- Serial(E), hold mode: $SH(E) = e_1 \wedge \neg E^n{}_2 \wedge \mathbf{X}(e_2 \wedge \neg E^n{}_3 \ldots \wedge \mathbf{X} e_n)\ldots)$
    - Each proposition in the sequence E becomes true in a specified order, one at each successive state, and all the next propositions are false at the moment. Once they become true, their true value does not matter.
    - $P_2 = R_1 \wedge \neg R_2 \wedge \neg R_3 \wedge \neg R_4 \wedge \mathbf{X}(R_2 \wedge \neg R_3 \wedge \neg R_4 \wedge \mathbf{X}(R_3 \wedge \neg R_4 \wedge \mathbf{X} R_4))$
- Serial(E), free mode: $\neg E \wedge \mathbf{X}\ SH(E)$
    - Each proposition in the sequence E becomes true in a specified order, one at each successive state. Once they become true, their true value does not matter.

- R_1 – конвейер 1 движется
- R_2 – датчик положения бутылки сработал
- R_3 – конвейер 1 стоит
- R_4 – клапан розлива открыт

SYSTEMS OF PATTERN CLASSIFICATION

(5) Complex events.

- Eventual(E), strict mode: $(e_1 \wedge \mathbf{XF}(e_2 \wedge \mathbf{XF}(e_3 \wedge \ldots \wedge \mathbf{XF} e_n)\ldots)$
  - Each proposition in the sequence E is asserted to hold in a specified order and in distinct and possibly nonconsecutive states.
  - $P_2 = R_1 \wedge \mathbf{XF}(R_2 \wedge \mathbf{XF}(R_3 \wedge \mathbf{XF} R_4))$
- Eventual(E), hold mode:

$$EH(E) = e_1 \wedge \neg E^n{}_2 \wedge (\neg E^n{}_2 \mathbf{U}(e_2 \wedge \neg E^n{}_3 \wedge (\neg E^n{}_3 \mathbf{U}(e_3 \wedge \ldots \wedge \mathbf{F} e_n)\ldots)))$$

  - Each proposition in the sequence E becomes true in a specified order and in distinct and possibly nonconsecutive states, and all the next propositions are false at the moment. Once they become true, their true value does not matter.
  - $P_2 = R_1 \wedge \neg R^4{}_2 \wedge (\neg R^4{}_2 \mathbf{U}(R_2 \wedge \neg R^4{}_3 \wedge (\neg R^4{}_3 \mathbf{U}(R_3 \wedge \mathbf{F} R_4))))$
- Eventual(E), free mode: $\neg E \wedge (\neg E \ \mathbf{U}\ EH(E))$

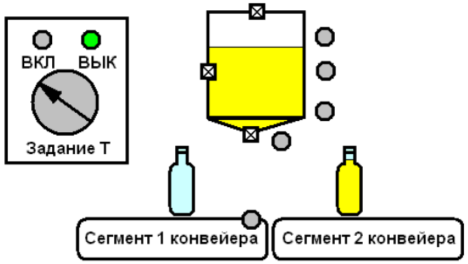
- P_0 – жидкость нагревается

SYSTEMS OF PATTERN CLASSIFICATION

(6) Quantity and Data.

- Quantity
  - Representation of common non-functional requirements
    - reliability (the number of errors in a given time window)
    - throughput (the number of requests that a client is allowed to submit in a given time window).
  - The exact, maximum, minimum, interval number of events
  - Point: $\mathbf{F}(P_0 \wedge \mathbf{XG}\neg P_0)$ – 1 time
  - Maximum: $\mathbf{F}(P_0 \wedge \mathbf{XG}\neg P_0) \vee \mathbf{F}(P_0 \wedge \mathbf{XF}(P_0 \wedge \mathbf{XG}\neg P_0))$ – at most 2 times
  - Minimum: $\mathbf{F}(P_0 \wedge \mathbf{XF}\ P_0)$ – at least 2 times
- Data
  - properties that refer to some data used in a system
    - every ID present in a message cannot appear in future messages.

SYSTEMS OF PATTERN CLASSIFICATION



(7) Environment.

Certain (undesirable) scenario of the behavior of the environment is really modeled.

- Bad behavior: **EF** Q $\wedge$ **AG** P
  - An environment can follow a bad pattern Q and a system always follows a good pattern P.

The property of optimal system behavior is that with any behavior of the environment, the system can achieve the desired state.

  - Expressible in μ-calculus.

- Optimality:
  - No matter how the environment behaves, the system can react so that the desired state P for the system and the state Q for the environment will be achieved.

PATTERN ONTOLOGY

	Operation	SubSpec1	SubSpec1
SimpleSpec	$\{\neg, \wedge, \vee, \rightarrow,$ $\leftrightarrow\}$	*Propositions* *SimpleSpec*	*Propositions* *SimpleSpec*
Spec		*Pattern* *Spec*	*Pattern* *Spec*

Specification

PATTERN ONTOLOGY

	Possibil	Frame Scope	Frame Time	Frame Quantity	Constraint	Kind	Atom1		
Pattern	bool	*Scope*	*Time*	*Quantity*	*Proposition*		*Proposition*		
Occurence						Absence Existence Universality			
Order						Precedence Responce		Atom2	Space
								Proposition	Novlap Rovlap Lovlap

Patterns

PATTERN ONTOLOGY

Data	Name1	Name2	Operation	Quantifier	Domain
	Names	Names	{>, <, =}	{∃, ∀}	Dom
Case	Kind	Name	Condition		
	state event	Names	*SimpleSpec*		
Compex	Kind	Type	Cases		
	strict free hold	one parallel serial eventual	*Case*		

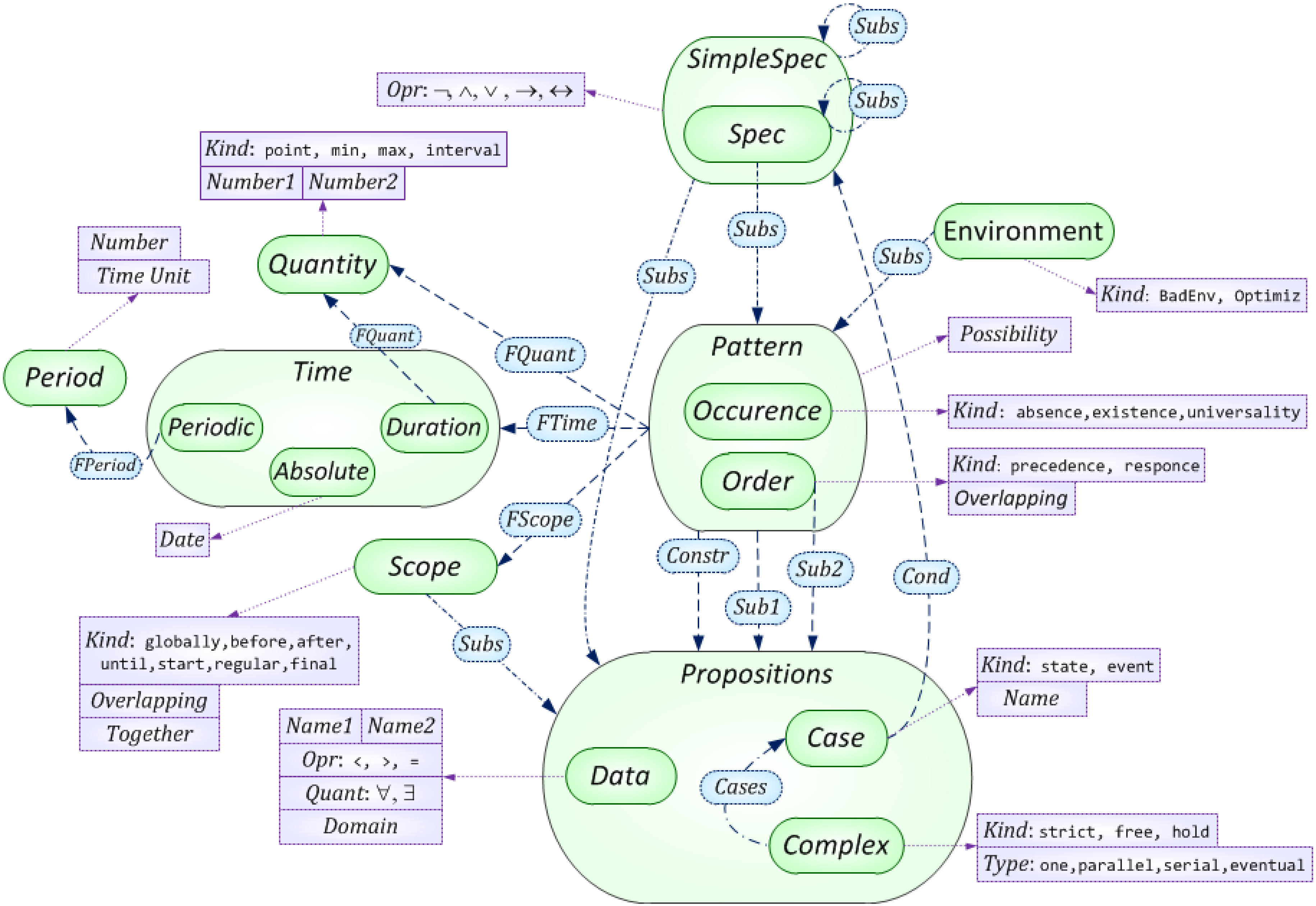
Propositions

PATTERN ONTOLOGY

<table>
<tr><td rowspan="2">Scope</td><td>Kind</td><td>Space</td><td>Together</td><td>Atom1</td><td>Atom2</td></tr>
<tr><td><code>globally, before after, ater-until start, regular, final</code></td><td><code>Overlapping</code></td><td><code>bool</code></td><td>Propositions</td><td>Propositions</td></tr>
<tr><td rowspan="2">Time</td><td>Kind</td><td>Frame Quantity</td><td>Frame Period</td><td>Calendar</td><td rowspan="6"></td></tr>
<tr><td><code>duration periodic absolute</code></td><td>Quantity</td><td>Period</td><td><code>Date</code></td></tr>
<tr><td rowspan="2">Quantity</td><td>Kind</td><td>Number1</td><td>Number2</td><td rowspan="4"></td></tr>
<tr><td><code>Point minimum maximum interval</code></td><td><code>Integer</code></td><td><code>Integer</code></td></tr>
<tr><td rowspan="2">Period</td><td>Time</td><td>Time Unit</td><td rowspan="2"></td></tr>
<tr><td><code>Integer</code></td><td><code>Time Domain</code></td></tr>
</table>

Bounds

PATTERN ONTOLOGY



CONCLUSION

- The first version of the ontology of requirements
  - qualitative, real-time, quantitative, taking into account combined events and statements about data.
  - In the future:
    - a generalization of patterns of event combinations to behavior
    - study specialized subject areas such as security, agent models, etc
      - specialized specification patterns.

CONCLUSION

- Construction and population the ontology-handbook of patterns
  - informal descriptions on a restricted natural language,
  - formal semantics in the language of modal logics
  - use cases
  - application scopes,
  - complexity of model checking and realizability, etc.
- Inconsistencies
  - forms of Until operator
    - inconsistence of the right time bounds of event occurrence.
  - the ambiguity of a natural language.
- Graphical formal languages (GIL,UML).

THANKS FOR YOUR ATTENTION!

EXAMPLE

- P_0 – жидкость нагревается
- P_1 – появление бутылки
- P_2 – наполнение бутылки
- P_3 – резервуар не переполнен
- Q_1 – система включена
- Q_2 – система выключена
- R_1 – конвейер 1 движется
- R_2 – датчик положения бутылки сработал
- R_3 – конвейер 1 стоит
- R_4 – клапан розлива открыт