CCNET- A SPECIFICATION LANGUAGE FOR MODELING CAUSALITY

Lazos Filippidis[1)], Holger Schlingloff[2)]
Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik FIRST
Address: Kekulèstr. 7, Berlin, Germany, D-12489,
E-mail: FiPublic@online.de[1)], hs@informatik.hu-berlin.de[2)]

Abstract: In the paper, the specification method **CCNet** is presented. The ontology, the signature and structure of the language of **CCNet** is given, i.e. its syntax and semantic. The language is based on the decomposition of natural language sentences and basically models causal relations by means of an information transfer between objects. **CCNet** is applied to an actual example.
Keywords:formal specification, natural language, causality

1. INTRODUCTION

Whenever malfunctioning of a system can cause extensive damage, the need for consistently and correctly specified systems exists. Therefore, relevant standards require methods and techniques to reduce the risk of hazardous system failures to an acceptable level of possible failure occurrence (IEC61508, 2003). For instance, to achieve a sufficient safety level, fault tolerance measures as e.g. hardware redundancy should be implemented.

However, the mechanisms of fault tolerance are compensated, if the system design is systematically incorrect. One reason arises from specification errors as a result of e.g. ambiguities or inherent design contradictions. In order to decrease the likelihood of failure occurrence due to design errors, a system has to be specified and tested properly. Elaborated specification and verification techniques aim at reducing design errors and allow tool-based model checking (Schneider, 2004).

But irrespectively of formal techniques, it is still necessary to specify and review a system specification on a top-level of system refinement by those persons, who are familiar with the required behavior of the system. Normally, the specification on top-level is based on natural language, which is understandable by different stakeholders. Therefore, a specification shall include modes of expression and descriptions which are understandable for the responsible personnel involved in the whole life cycle of the system. The common basis of communication between differently trained specialists seems to be natural language, although it is known, that it lacks unambiguity and completeness. However, that informal type of meaning representation represents the current state-of-the-art of requirements and systems specification.

To narrow the gap between natural language and formal methods, we propose the specification language **CCNet** (Causal-Context Net). This language is based on meaning representation and provides a formal semantic and syntactic layer. It decomposes natural language sentences revealing the causal deep structure within a given context. The included causality semantic provides a framework for reasoning about causes and their effects within a given context.

1.2 Related works

Starting with Hume's regularity theory, a huge number of proposals to model and reason about causality has been proposed in the past (Pearl, 2000).

Decomposing natural language formally was tried in (Jackendoff, 1990) and (Helbig, 2000), but without explaining the semantics of causality.

Most of the formal techniques applied to technical systems focus on state-event based modelling using temporal logic, whereas the meaning of causality of the state changes ist not taken into account.

In (Bitsch, 2007), safety patterns were given, which concentrate on temporal aspects. However, these patterns do not analyze the semantic structure of the safety design.

Modelling causal dependencies regarding system failure analysis, to the best of our knowledge as a formal technique only WBA (Why-Because-Analysis) has been used (Ladkin, 2001). WBA is based on Lewis' counterfactual interpretation of causality considering the closest word assumption (Lewis, 1973). This method is not related to natural language meaning representation.

Our approach aims at specifying causal structures within a system, which should be closed as possible to natural meaning representation, in order to reason about the behaviour of a system. Our approach is based on a modified type of the relatively new or rather newly discovered transference theory of causality due to its intuitive nearness to physical and technical systems (Dowe, 2000) (Kistler, 2002). We assume, that - within a causal field (Mackie, 1980) - a technical system can be decomposed into atomic causal relations between states, whereas information is transferred between entities.

2. SYNTAX UND SEMANTICS OF CCNet

We define the signature and structure of **CCNet** and connect them with natural language meaning representation. We restrict the language **CCNet** to a subset of possible words which we call physical word.

Definition 2.1 (Physical Worlds) *The physical world P contains all worlds p out of possible worlds, in which the currently known physical laws hold. A physical world of discourse p' is an element of P.* □

2.1 SYNTAX OF CCNet

Definition 2.2 (Possible Words and Sentences of CCNet) *Let $w \in \Sigma^W$ an arbitrary sequence of characters out of the alphabet Σ={a, b, … z, 1, 0, 2, …9,-} and Σ^W the set of all possible sequence of characters. w is termed a word out of Σ^W.*
An arbitrary concatenation of words $s' \rightarrow s^\circ\_w'$ with concatenation symbol '_', a starting word $w \in \Sigma^W$ with $s^0 \rightarrow w^\circ\_w'$, $w' \in \Sigma^W$, $\{s^0, s, s'\} \in \Sigma^S$ is termed a sentence (italic written), Σ^S the set of sentences. □

We connect natural language and physical worlds *p*, resulting in a formal language and being a subset of the union set $\Sigma^W \cup \Sigma^S$. But before, in order to define well-formed words and sentences, we define the entities and entity types.

Definition 2.3 (Entity types, entities and atomic entities of CCNet)
- *Entity types [et] are sets containing entities.*
- *Elements of the sets are termed entities |ent> and represent all facts or objects of a physical world, about which one can make statements.* □

The next two definitions give the signature and structure of **CCNet**. We define both a textual and graphical syntax for elements of the language.

Definition 2.4 (Signature of CCNet)
*The signature of **CCNet** is a triple $S_{CCNet}=(D,R,F)$ with:*
- *D is a non-empty set of entity types $[et] \in D$.*
- *R is a non empty set of binary relations $REL \in R$: $REL: \langle et_1, et_2 \rangle$; $[et_1], [et_2] \in D$.*
- *F is a non-empty set of functions $*FUNC \in F$: $*FUNC:[et_1]\ldots[et_n] \longrightarrow [et]$; $[et],\ldots, [et_n] \in D$.* □

The structure of **CCNet** is given by the following definition:

Definition 2.5 (Structure for CCNet) *Let $S_{CCNet}=(D,R,F)$ be the signature of **CCNet**. The S_{CCNet}-structure over the signature S_{CCNet} is a triple $<A_{CCNet}=((A_{[et]})_{[et]\in D}, (*FUNC_A)_{*FUNC\in F}, (REL_A)_{REL\in R})>$ with:*
- *For each $[et] \in D$ there exits a non-empty set $A_{[et]}$ with elements $|ent> \in [et]$.*
- *For each binary relation symbol $REL_A \in R$ there exists a relation REL_A with $REL_A \subset A_{[et1]} \times A_{[et2]} \subset A_{[et]}$.*
- *For each n-ary function symbol $*FUNC_A \in F$ there exists a function $*FUNC$ with $*FUNC:A_{[et1]} \times \ldots \times A_{[etn]} \longrightarrow A_{[et]}$.*

$A_{[et]}$, REL_A, $*FUNC_A$ are written [et], REL and FUNC. Entity types and their subsets are enclosed by [ ]. □

The textual and graphical syntax of **CCNet** is based on the representation formalism of semantic networks (Helbig, 2001). In principle,

the network contains node symbols for entities and directed arcs for relations and functions, which connect entities and in turn reveal new entities.

Definition 2.6 **(Syntax of CCNet)**
Let $S_{CCNet}=(D,R,F)$ be the signature of ***CCNet****.*

- *';' ',' '(' ')' '[' ']' '=' '?' '!' '<>' '=>' '=' '<=' '<' '>' are symbols.*
- *An entity (|ent>) is a formula and is graphically represented by a node symbol (filled circle).*
- *(|?ent>) is an entity.*
- *(|!ent>) is an entity.*
- *(|ent>)=(<ent₁|REL|ent₂>), REL∈R, is a formula and is graphically represented by an arrow between the related entities |ent₁> and |ent₂> with an open circle as a new node symbol for (|ent>).*
- *(|ent>)=(|*FUNC(|ent₁>,…, |entₙ>)>), *FUNC∈F, is a formula and is graphically represented by a bundle of arrows directed to the co-domain |ent>. (*FUNC(.)) and (|*FUNC(.)>) is used synonymously.*
- *A formula (|form>) is an entity and is recursively defined. If form₁, form₂ are formulae, then (|form>)=(|form₁>; (|form₂>); is a formula. Graphically a formula is enclosed by a dotted-bordered area with an entity symbol for (|form>).*
- *A set of formulae is a system* ***S****. A graphical representation of a system* ***S*** *is named a semantic network.* □

If it is clear by context, we suppress the parentheses, e.g. (|ent>) is substituted by |ent>. The parentheses are used as usual.

Definition 2.7 **(Node label of CCNet)** *Let the signature and structure of* ***CCNet*** *be defined and |ent>∈[et].*

Node labels $nl_1,\ldots, nl_n$ with values $vl_1,\ldots, vl_n$, written (|ent>[nl_1@vl_1,…,nl_n@vl_n]) with the symbol @ standing for the symbols {>,<,=>,=<,<>,=} can be assigned to entities. □

In order to illustrate both the textual and graphical syntax, in (Figure 1) we give some examples. |a> represents any entity, graphically symbolized by a circle. The relation <a|R|b> connects |a> and |b> yielding a new entity, graphically symbolized by an arrow with an open circle. The function *F maps the two entities |a> and |b> to the entity |c>, which is graphically symbolized by two connected arrows.
These elements may be combined in a more complex specification: |d> is related to |a> by the relation Q. |a> and |b> are mapped to |c> by the function *F, which is related to |e> by the relation R. |x> is a formula comprising the enclosed elements.

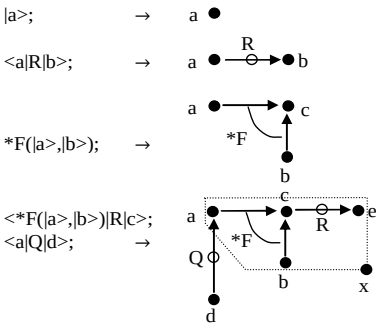


Fig.1. Examples of both textual and graphical syntax of CCNet.

2.2 SEMANTICS OF CCNET

Beside their existence and non-emptiness of the sets of entity types, so far nothing is said about the carrier sets themselves. On the basis of the ontology of natural language provided in (Helbig, 2001), we give a definition of entity types by means of its relational or functional use.

Before we start defining entity types, we define the symbols '!', ';' and '?' semantically.

Definition 2.8 **(Substitute '!', Unspecified '?')**
Let $w\in\Sigma^w$ be a word.

- *An entity |!w> is a substitute for an entity |x> termed by |!w>=|x>.*
- *An entity |?w> is a not further specified entity of a given entity type.* □

Definition 2.9 **(Observable Formula)**

- *If formulae |form₁> and |form₂> hold, then |form₁>; |form₂> conjunctively hold.*
- *A formula holds, if it is observable in the physical world of discourse p'.* □

In the following, the ontology of **CCNet** is defined. The ontology underlies the defined

structure: Objects and facts of the physical world are mapped to **CCNet** entities.

Definition 2.10 **(Object [o])** *The carrier set [o] contains all entities of a stable physical or mental characteristic within space-time, which might be linked to each other. [o] is the union set of [op]=[opm]∪[opnm]∪[opca] and [om]=[omq]∪[omab] with disjoint sets:*

- *[opm] not-animated physical objects with a measurable extension in space-time and mass density.*
- *[opnm] not-animated physical objects with a measurable extension in space-time without a mass density with {|time>, |space>}∈[opnc].*
- *[opa] animated physical object with a measurable extension in space-time and mass density.*
- *[omab] abstract object standing for an arrangement or constellation of objects.*
- *[omq] abstract object, which are a quantitative or qualitative attribute or property, associable to an physical object.*

Elements of [o] are specified by a word of natural language and are associated to nouns. □

The next two definitions give the central entities of situations, called processes, and situation carrier.

Definition 2.11 **(Process [pr])** *The carrier set process [pr] contains all entities, which reflect a constellation of objects or their behavior within space-time within a physical world. [pr] is the union set of the disjoint entity types event [ev] and state [st] with* $\cup_{i\in\mathbb{N}}(|ev_i>; st_i>)\in[pr]$, $|st_i>\in[st]$, $|ev_i>\in[ev]$ $(i\in\mathbb{N})$*:*

- *[st]: state of constellation of objects or being of object within space time without timely change (state) and with* $\cup_{i\in\mathbb{N}}|st_i>\in[st]$, $|st_i>\in[st]$.
- *[ev]: change of constellation of objects or change of object within space (event) and with* $\cup_{i\in\mathbb{N}}|ev_i>\in[ev]$, $|ev>\in[ev]$.

A process is also called a situation. □

Definition 2.12 **(Situation Carrier [sic])** *The carrier set [sic] contains all entities, which determine ('carry') the constellation of an object in respect to other objects within space-time.*

Elements of [sic] are specified by a word of natural language and are associated to verbs. □

If a process has no resulting state, it is a state. If the resulting state is specified by a change relation, a process is an event. Therefore, it is possible to integrate time without the distinction between telic and atelic verb phrases, i.e. situations are always atelic as long as the resulting states are not specified.

Definition 2.13 **(Situation specifier [ss])** *The carrier set [ss] contains all entities, which determines the context of an object or process. [ss] is the union of [sslc], [sstmp] and [sspr], where*

- *[sslc]=[sslcl]∪[sslcp] is a local specification of a process or an object. [sslcp] specifies a point location and [sslcd] a direction.*
- *[sstp]=[sstpp]∪[sstpd] is a time specification of a process or an object. [sstpp] specifies a time point and [sstpd] a time duration.*
- *[sspr]={|probability>, |rate>} specifies the occurrence rate or probability of a process.* □

We give some examples to illustrate the use of the situation specifier. For that, we use the functions **dist(,,)* for the generation of a local distance, **above(.)* for indicating, that something is above something, and the relation LOC for locating a process or an object at a locality. The relations EXP describes an experiencing relation, *timedist (,,) the time distance function and TIME the time relation. CHNL indicates a change of a node label.

Example 2.1 ***(Location Specifier)***

- *'The distance between a and b is 20 m':* **dist(|a>, |b>)=20_m;*
- *'20 m above a':* *<b|LOC|*above(|a>)>; *dist(|a>, |b>)=20_m>;*

Example 2.2 ***(Time Specifier)***

- *'a starts after 20 time ticks':* *|!x>=<<a[valid=no]|EXP|start>|CHNL|a[valid =yes]>;* **timedist(|!x>, |?b>)=20_timeticks;* *<!x|TIME|*before(|?b>);*

In Example 2.2, the use of the unspecified entity |?b> is necessary, in order to define the time distance between the starting of a fictive process and the occurrence of |a> (see Figure 2).

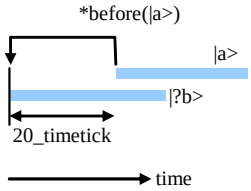


Fig.2. Examples of both textual and graphical syntax of CCNet.

The specification language **CCNet** avoids imprecise use of local or temporal phrases not allowing adverbial phrases like '*here*', '*there*' or '*now*', '*yesterday*'.

An attribute or a situation specifier may have a qualitative or quantitative value. The following entity types consider that aspect.

Definition 2.14 **(Quality and Quantities)**
The carrier set [q] is the union of [qlv], [qn] where

- *[qlv] is a qualitative value of an attribute.*
- *[qn] is quantity value of an attribute.*

The entity type [qn] is substituted by the types [qnv] and [qnu], the value and the unit of the quantity. □

2.3 NODE LABEL SEMANTICS

Adopted from (Helbig, 2001), the language of **CCNet** assigns node labels to entities. Node labels are required for further specification of entities on a meta-level. The following definition gives their semantic and uses the Definition 2.7.

Definition 2.15 **(Node Labels)**
Let the signature and structure of ***CCNet*** *be given. The semantics of node labels are as follows:*

- *The node label 'valid' can be assigned to entities of entity type [o], [sic], [ss], and [pr] with:*
  - *The default label 'valid=yes' assigns an entity an observable existence in a physical world.*
  - *The label 'valid=no' assigns an entity an observable non-existence in a physical world.*
  - *The label 'valid=hypo yes' assigns an entity a hypothetical observable existence in a physical world.*
  - *The label 'valid=hypo no' assigns an entity a hypothetical observable non-existence in a physical world.*
- *The node label quant can be assigned to entities of entity type [o]\[omq] with (m, n* $\in \mathbb{N}_0$*).*
  - *default 'quant=all' (all elements of referenced set of entities).*
  - *'quant=n' (exactly n elements of referenced set of entities).*
  - *'quant=(>=m | n)' (more than/ equal m out of n entities referenced set of entities).*
  - *and further.*
- *The node label time can be assigned to entities of entity type [pr] with (n* $\in \mathbb{N}_0$*):*
  - *'time=n' (n times of occurrence, default n=1)*
  - *'time<>n' (not n times of occurrence).*
  - *and further.*
- *The node label ref can be assigned to entities to entities of entity type [o]\[omq], [ss], and [pr] with:*
  - *ref* $\in \Sigma^W$ *and 'ref=ref' a unique reference to elements of entities within the physical word of discourse.*

  If the node label ref is not assigned, the entity references to all possible entities of the physical world of discourse.

Node labels of different types can be conjunctively assigned to an entity. □

We conclude the definitions of the syntax and semantics of **CCNet** with an example, where node labels are used.

Example 2.3 ***(Node label time)***
'a occurs less than n times in time intervals of 20 ms'
|!x>=<<a[valid=no]|EXP|occur>
|CHNL|a[valid=yes]>;
|!x'>=<<a[valid=yes]|EXP|occur>
|CHNL|a[valid=no]>;
**timedist(|!x>, |!x'>)=20_ ms;*
*<!x|TIME|*before(|!x'>)>[time<n];*

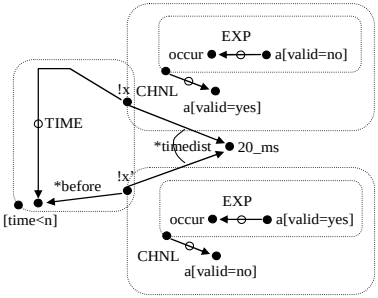


Fig.3. Example for the use of node labels.

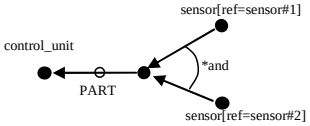
3 EXAMPLE – A REDUNDANT SENSOR SYSTEM FOR SWITCHING ON A PROTECTION

In order to show the expressiveness of **CCNet**, we give a fictive example of a control unit of a fuel tank filled with gas. If the internal gas pressure exceeds a critical level, a protection mechanism shall reduce the gas pressure. To achieve an acceptable low risk level, we assume, that two sensors, which measure the gas pressure, are redundantly connected (2oo2 system). The redundantly connected sensors are part of the control unit. Furthermore, a micro controller shall receive the output of both sensors. In emergency case, the micro controller initiates a process for reducing the gas pressure (e.g. by opening a gas valve).

In the following example, the underlined words reflect the natural language representation. The relations and functions become clear by comparing the natural language sentences and the graphical representation. Beside the relations defined above, PART describes a part relation, CTXT a context relation, TR a transmit relation and CST a causing relation. The symbols '->' and '~' stand for '*containing information*' and '*in occurrence of*', respectively. The functions are self-explanatory.

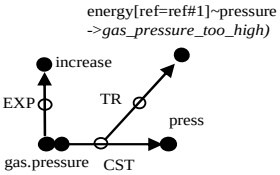
The example is reduced step by step:

- *state(1)*:
  <*and(sensor[ref=sensor#1],
  sensor[ref=sensor#2])|PART|control_unit>;



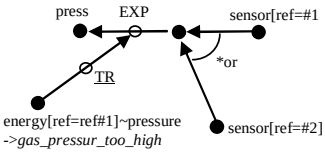
The sensor referenced by sensor#1 and the sensor referenced by sensor#2 are part of the control unit.

- *process(2)*:
  <gas.pressure|EXP|increase>;
  <<gas.pressure|CST|press>
  |TR|energy[ref=ref#1]~pressure->
  gas_pressure_too_high>;



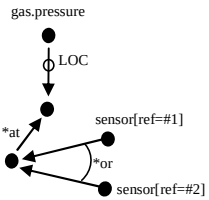
The gas pressure (pressure is an attribute of the gas) experiences an increasing. Due to the state change of the gas pressure (not presented here) the gas pressure causes a pressing by transmitting pressure; the transmitted energy contains the information content 'the pressure is too high'.

- *process(3)*:
  <energy[ref=ref#1]~pressure->
  gas_pressure_too_high|TR|
  <*or(sensor[ref=sensor#1],
  sensor[ref=sensor#2])|EXP|press>>



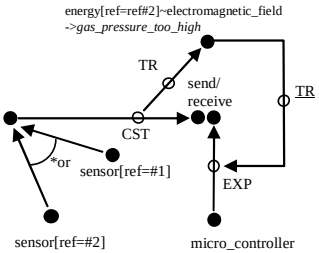
The sensor referenced by sensor#1 or the sensor referenced by sensor#2 experience a pressing by means of the transmitted gas pressure; the transmitted energy contains the information content 'the pressure is too high'.

- *process(4)*:
  <!c|CTXT|(<gas.pressure|LOC|
  *at(*or(sensor[ref=sensor#1],
  sensor[ref=sensor#2])>)>



The causation process (process(2) and process(3) referenced by the unspecified entity |!c>) happen in the context, that the gas pressure is at the sensor referenced by sensor#1 or at the sensor referenced by sensor#2.

- *process(5)*:
<<*or(sensor[ref=sensor#1], sensor[ref=sensor#2])|CST|send>|TR| energy[ref=ref#2]~electromagnetic_field *->gas pressure_too_high>;*
<energy[ref=ref#2]~electromagnetic_field *->gas pressure_too_high*|TR|
<micro_controller|EXP|receive>>



The sensor referenced by sensor#1 or the sensor referenced by sensor#2 cause a sending and transmit the electromagnetic field containing the information content 'gas pressure is too high'; the micro controller experiences a receiving by means of the transmitted electromagnetic field containing the information content 'the gas pressure is too high' from the sensor referenced by sensor#1 or the sensor referenced by sensor#2.

Combining the specified processes (1)-(5) yield the following network of Fig. 4.

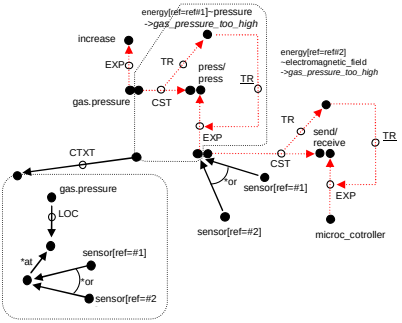


Fig. 4. Representation of the measurement of gas pressure detected by a 2oo2 sensor system. The dotted arrows reflect the causal flow. Natural language description is marked italic.

Some further remarks may be helpful: Firstly, temporal aspects are suppressed in the graphical representation of the causal network. They are implicitly specified by means of the path of the causal process, which can be tracked by following the CST/EXP (causing/ experiencing) and TR/TR (transmitting/ transmitted) relations (indicated by the dotted arrows).

Secondly, the entity types of the entities are not explicitly given in the examples. Implicitly, they are specified by their relational use. A complete definition may be given by implementing a lexicon containing the used entities and their corresponding entity types. The lexicon can be used for checking the correct use of the entities.

4 CONCLUSION AND OUTLOOK

In this paper we have defined the formal language **CCNet** by means of its syntax and semantics. One of the main advantages of **CCNet** is the strong connection to natural language. We use a minimal, but sufficient set of entities, relations and functions for domain-specific representation of facts or situations regarding technical systems.

We have defined some more relations and functions beside the given ones, which model standard safety architecture patterns like supervision or barrier functionalities.

One of the central features of **CCNet** is the integration of causality by means of an entity transfer. Based on that, currently a calculus is

developed for reasoning about failure behavior. The calculus is based on counterfactual reasoning (Pearl, 2000).

References

Bitsch, F. (2007). Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme in Temporallogik. *Aachen: Shaker*.

IEC 61508 (2005). Functional safety of electrical/electronic/programmable electronic safety-related systems.

Helbig, H. (2001) Die semantische Struktur der Sprache. *Heidelberg: Springer*.

Jackendorff, R. (1990). Semantic Structures. *Mass.: MIT Press*.

Dowe, P. (2000). Physical causation. Cambridge: *Cambridge University Press*.

Kistler, M. (2002): Erklärung und Kausalität. *Philosophica Naturalis 39, Heft 1*, 89-109.

Ladkin, P.B. (2001). Causal System Analysis-Formal Reasoning About Safety and Failures. *Heidelberg: Springer*.

Lewis, D. (1973). Counterfactuals. *Oxford: Blackwell*.

Mackie, J. L. (1980). The Cement of the Universe: A Study of Causation. Oxford: *Oxford University Press* (reprint of paper back edition, 2001).

Pearl, J. (2000). Causality. *Cambridge: Cambridge University Press*.

Schneider, K. (2004). Verification of Reactive Systems. *Berlin: Springer*.