Appeared in: 12th International Workshop on Principles of Diagnosis, 2001.

G^+DE - The Generalized Diagnosis Engine

Ulrich Heller[1] and Peter Struss[1,2]

[1]Model-based Systems and Qualitative Reasoning Group of the Technical University of Munich,
Orleansstr. 34, 81667 Munich, Germany
[2]OCC'M Software GmbH, Gleissentalstr. 22, 82041 Deisenhofen, Germany
heller@in.tum.de, struss@in.tum.de

Abstract

The existing theory of consistency-based diagnosis and its implementations have proven successful in a number of technical applications. However, they turn out to be inherently limited to a very specific class of systems to be diagnosed: They are tailored for artifacts consisting of components in a fixed structure, and they are aimed at a particular kind of diagnosis and repair, namely failing components and their replacement.
In order to cover systems that comprise processes and change their structure dynamically, faults that are due to unanticipated objects and interactions, and therapy tasks that involve complex interference with the system, we need an extended theory of diagnosis and therapy. In this paper, we present theoretical and practical results of our work on process-oriented consistency-based diagnosis.
For this purpose, a logical reconstruction of a process-oriented modeling paradigm is needed, which forms the foundation for both the automated composition and the revision of system models to be used in checking consistency. It enables a concise characterization of solutions to the tasks of situation assessment and therapy recognition in a non-monotonic logic formalism.
We outline an algorithmic approach to computing the specified solutions, by transforming the non-monotonic theory into monotonic composition plus constraint-based consistency checking, potentially followed by a search for structural revisions. A key step in this is the explicit representation of the closed world assumption during model composition and its revision. These algorithms form the basis of an implemented prototype, the Generalized Diagnosis Engine (G^+DE).

Introduction

Traditional consistency-based diagnosis is seen as a search for behavior mode assignments to the components C_i of a system, such that this assignment is consistent with the respective system model (or system description, SD) and the existing observations of the system behavior, OBS:

$$SD \cup OBS \cup \{mode_i(C_i) \mid C_i \in COMPS\} \not\vdash \bot .$$

In [Struss/Heller 1998], we argued that this approach (underying [de Kleer/Williams 1987], [Reiter 87], [de Kleer/Williams 1989], [Struss/Dressler 1989], [Dressler/Struss 1994] etc.) is fairly restrictive because it is based on a number of (implicit) assumptions about the type of systems, faults, and diagnosis goals:

- The system is composed of known components,
- has a known fixed structure
- and is well-designed, i.e. exhibits a proper behavior when all components behave normally.
- All faults are due to failures of existing components, and, in particular,
- no structural faults occur.

Moreover, it is often taken for granted that localizing a fault in a component is a natural goal of diagnosis which is not necessarily true if the remedial action is different from component replacement.
In response to this, we proposed to develop a theory and diagnosis system that addresses

- systems with a dynamic (process) structure,
- faults caused by unanticipated objects and interactions,
- explicitly stated behavior goals, and
- complex therapy measures.

In this paper, we present G^+DE, the Generalized Diagnosis Engine, our first answer to this challenge. It supports two different tasks:

The first task is to answer the question "What's going on?", and observations are used to answer it. We call this task "Situation Assessment" and define it as finding a system description, SD', that is consistent with the observations, by revising an initial system description, SD, that is not:

Situation Assessment:

$$SD \cup OBS \vdash \bot \quad \Rightarrow \quad SD' \cup OBS \not\vdash \bot$$

The result of this step, SD', contains a structural description and quantity value assignments (which is more general than mode assignments, as in the classical case) and provides us with a hypothesis about the state of the system under consideration. In contrast to the classical consistency-based component-oriented diagnosis, SD' cannot be expected to directly answer the question "What's going wrong?" (namely by enumerating which components are broken) or "What can be done?" (namely by replacing the broken components). But it will be used as the basis for determining a "treatment".
Therapy recognition is the second task, which can also be

described as a revision of a system description (usually a result of situation assessment) inconsistent with a criterion into a consistent one:

Therapy Recognition:

$$SD' \cup GOALS \vdash \perp \quad \Rightarrow \quad SD'' \cup GOALS \nvdash \perp$$

Here, the migration from SD' to SD" is achieved by triggering behavior constituents as a result of changes of the structure or quantities that can be controlled by an agent, i.e. therapy actions. Although it is important to distinguish these tasks, it is obvious that there is a common characterization of both tasks:

Common Characterization of Tasks:

Apply minimal changes to a given initial model in order to achieve consistency with a given criterion (observations or goals, respectively).

We continue by presenting an example from our application domain, water treatment. The following sections describe the modeling formalism and characterize the solutions to the above tasks as (minimal) consistent completions of initial partial models. Finally, we discuss the computation of these solutions and its foundation and present the architecture of $G^{+}DE$.

A Motivating Example

The following example is a simplified scenario from a collaborative project in the domain of hydro-ecology and water treatment carried out with partners in Porto Alegre, Brazil. In summer days, a distinctly unpleasant metallic taste was observed in the drinking water processed from raw water captured in a small reservoir. Analysis of water samples confirmed a high concentration of dissolved iron. However, there was no known source of iron - neither in the treatment plant nor in the ecosystem itself. The situation is shown in a simple diagram in Figure 1.

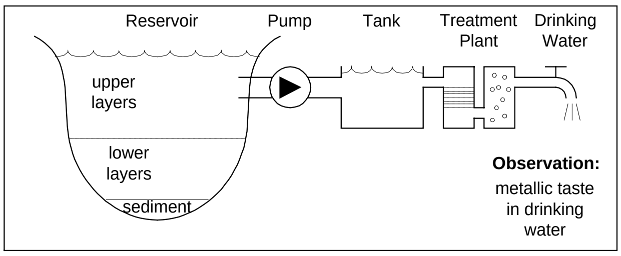

Figure 1 Example scenario - observation of metallic taste

Environmental experts came up with a surprising hypothesis: There is a high amount of solid iron in the sediment, which was unknown so far. The pH of the bottom layers (hypolimnion) typically lies in a medium range, and this has almost completely prevented the redissolving of iron into the water body. Suddenly, however, the pH has been significantly lowered, the most likely cause being a local algal bloom triggered by excessive accumulations of nutrients. The iron dissolved, ascended to the surface layers, was captured with the raw water intake, and the standard treatment process could not prevent it from reaching the drinking water.

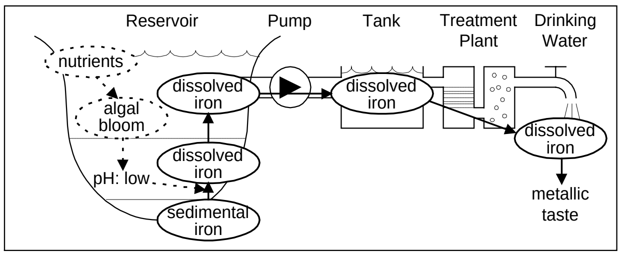

Figure 2 Example - situation assessment

After confirming this situation assessment (depicted in Figure 2), experts discussed possible countermeasures. Certainly, removal of the newly discovered sedimental iron was not an option. Viable alternatives included adding an oxidation agent in the early stages of the treatment process and artificial alkalization in the water body. For future occurrences of local algal blooms, algaecides and long term plans for the prevention of eutrophication were considered. All alternatives are shown in Figure 3.

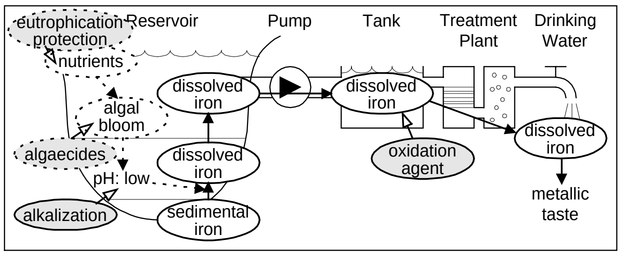

Figure 3 Example - possible countermeasures ("therapy recognition")

The example presented above clearly describes a diagnostic task in the sense of situation assessment and therapy recognition. However, the "classical" form of consistency-based diagnosis fails to meet the requirements as put forth by a large class of problem domains, an instance of which is demonstrated in the example.

For a complete treatment of this example see [Heller 2001], where also a number of extended application examples from different domains can be found.

The Modeling Approach

In this section, we briefly summarize the foundation of our modeling approach which is basically a logical reconstruction of a relevant part of *Qualitative Process Theory* ([Forbus 1984]) and has already been described in ([Struss/Heller 1998]).

The Modeling Language

The basic building blocks of the modeling language are the Domain Theory and the Situation Description (Figure 4).

The *Domain Theory* consists of an Ontology, Quantities, and a set of Behavior Constituent Types, as well as the Basic Axioms. The *Ontology* defines the entities that can be used in representing structure: object types and structural relations. *Quantities* of defined types can be associated with object instances. *Behavior Constituent Types* are an abstraction of process and component behavior models. They are the fundamental elements of behavioral descriptions in deterministic laws of the form:

IF *Structural Conditions* AND *Quantity Conditions*
THEN *Structural Effects* AND *Quantity Effects*.

Semi-formally, we have the following occurrence rule:

(OR) $SC \wedge QC \rightarrow SE \wedge QE$

The quantity effects comprise constraints and influences on variables that are local to the behavior constituent and the objects mentioned in the structural conditions and effects.

The *Basic Axioms* are the formal representation of the semantics of the modeling primitives, e. g. the laws governing type hierarchies and quantity values assignments, or the occurrence rule for behavior constituents. Finally, a *Situation Description* is similar to what QPT calls a scenario. It consists of object instances (and relationships between them) and initial values of associated quantities, all possibly supported by retractable user-defined assumptions.

Figure 4 gives an overview of the model structure and also indicates the specificity of the modeling sections.

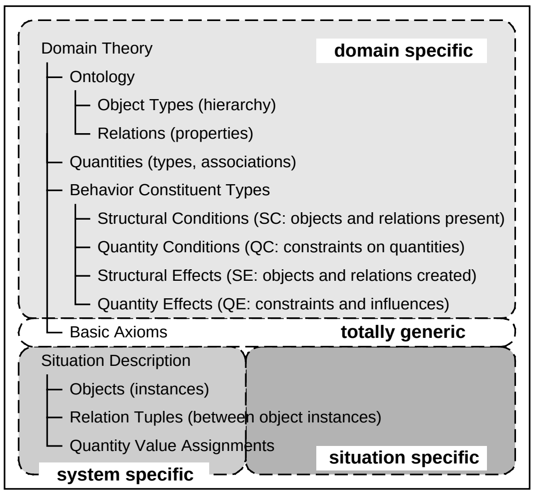
Domain Theory
domain specific
Ontology
Object Types (hierarchy)
Relations (properties)
Quantities (types, associations)
Behavior Constituent Types
Structural Conditions (SC: objects and relations present)
Quantity Conditions (QC: constraints on quantities)
Structural Effects (SE: objects and relations created)
Quantity Effects (QE: constraints and influences)
Basic Axioms
totally generic
Situation Description
Objects (instances)
Relation Tuples (between object instances)
Quantity Value Assignments
situation specific
system specific

Figure 4 The Modeling Sections and their Specificity

As an example, Figure 5 shows the process of redissolving of iron. It is stated in a graphical notation which can be manipulated by an editor included in G$^+$DE. It indicates

- Structural conditions (plain boxes with double lines and circles with arrows): solid iron fixed in the sediment which is below a water layer,
- Quantity conditions (small shaded boxes with values restrictions attached): the pH being negative (i.e. acidic) and the concentration of the sedimental iron greater than zero,
- Structural effects (boxes and circles with stars): iron dissolved in the water layer, and
- Quantity effects (rounded boxes and arrows with a sign): the (qualitative) DIFF-constraint expressing the dependency of the newly created variable rate on pH and concentration, and influences of this rate on the iron concentration in the sediment and the water layer.

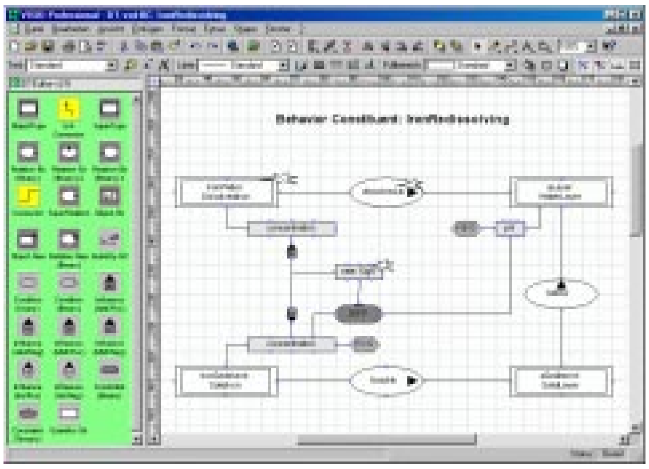

Figure 5 The process of iron redissolving in the graphical notation of G$^+$DE's domain theory editor

Closure of the Domain Theory

An important part of the *Basic Axioms* are closure axioms. Firstly, the domain theory is closed w.r.t. types. Formally, this can be expressed as a set of static closure axioms.

Secondly, there are closure axioms for predicates the extension of which is not known from the beginning. Most importantly, any influence has to arise from an occurring behavior constituent. The application of this axiom on derivatives comes close to the "sole mechanism assumption" in QPT which dictates that no value change can occur except as an effect of a process.

Thus, the domain theory is both closed and fixed - while the situation description (comprising system and situation specific parts, see figure 4 above) can be completed and partially changed or revised. The next section will discuss this in more detail.

Situation Assessment and Therapy Recognition - Characterizing Solutions

Completing Partial Models

The proposed modeling language does not allow for negative propositions in the situation description: everything that is not mentioned is not there - except if it has to be. There are two ways, an additional object or relation can be identified as necessary and, consequently, be introduced into the situation description:

- *Forward completion*: the given situation description could trigger a behavior constituent, which will create the additional structural element, thus completing the model in a (causal) forward direction.
- *Backward completion*: effects could be observed that cannot be attributed to the situation description and its forward completion alone, but rather point at causes not yet considered.

We will characterize solutions based on a minimization of such completions (of both kinds) - usually yielding multiple possibilities.

Characterization of Solutions:
Minimal consistent completion of the initial situation description fulfilling a maximal set of user-defined assumptions.

While the classical approach to consistency-based diagnosis can handle explicitly included assumptions well (such as behavior mode assumptions for components), changes in the structure - as effects of occurring behavior constituents or concluded from their effects - are novel to the generalized approach presented here.

Three sets of predicates are used to distinguish structural elements based on their justification. Objects and relation tuples introduced by the user, i. e. included in the initial situation description, are marked with KnownElement(e). The initial situation description is closed with respect to these predicates in the formal sense.

The "forward" completion comprises the objects and relation tuples from structural effects of necessarily occurring behavior constituents, which is expressed by the predicate EffectElement(e). These elements are required to be supported by occurring behavior constituents (for formal semantics please refer to [Heller 2001]).

For the "backward" completion, i. e. structural elements that are deduced to be part of the model from their respective effects, we use the predicate Introduced-Element(e). The minimization semantics will rely on these predicates. But they are also the key to ensure that only structural elements that are specified as "introducible" can occur as the primary causes in the backward completion.

Minimality of Completions

We require that only necessary elements are added (in both kinds of completions) to the ones explicitly named by the user. As a first step, we exclude any structural elements that are not contained in any of the defined classes:

$$\text{Element(e)} \rightarrow \text{KnownElement(e)} \vee \text{EffectElement(e)} \vee \text{IntroducedElement(e)}$$

Additionally, the respective predicates will be minimized. For the initial situation description this is achieved by static closure (as defined above) and for the forward completion (EffectElement) this is ensured by the definition requiring support by an occurring behavior constituent.

The critical part is backward completion: We define a semantics based on minimization of the predicate IntroducedElement. A structural addition is minimal, if no proper subset achieves consistency. It is important to note that this cannot be expressed in first-order logic and we resort to prerequisite-free normal default schemata (cmp. [Reiter 1980]):

(DE) $\neg$ IntroducedElement(e) / $\neg$ IntroducedElement(e)

with the intuitive reading: "if an object or relation tuple is not necessarily present (i. e. its absence is consistent), then it is indeed absent". Adding these schemata to the theory defined by the domain theory, the situation description and all discussed static closures, will yield exactly the minimal completions as extensions - modulo contradictory user-defined assumptions.

Retraction of User-defined Assumptions

There is a second criterion for preferring one candidate solution over another, namely that user-defined assumptions have to hold whenever possible. In default logic, we state that we do not want to retract user-defined assumptions without necessity:

(DA) Holds(assm) / Holds(assm)

Of course, both kinds of revisions do interact with each other, since the completions are to be understood w. r. t. the part of the situation description that is not retracted.

The given characterization of solutions as extensions is concise, but does not provide an effective method for calculating actual extensions for a given theory. The domain theory contains all information defining the solutions, but for actually computing the completions, a sophisticated approach is required.

Computing Solutions

The Key Idea

From an abstract perspective, our approach is to compute the forward completion, as characterized above - and all occurring behavior constituents resulting from it - and then to use the GDE algorithm to simultaneously check its completeness and the consistency of the set of user-defined assumptions. In the case of inconsistencies, retraction of user-defined assumptions and/or backward completion of the structure are necessary to achieve valid solutions.

Figure 6 shows how a structure initially specified in the situation description as provided by the user (shaded middle part) can be completed both by necessary effects (to the right) and by necessary additional causes (to the left of the situation description). The extensions of the predicates used in defining the solution are indicated (KnownElement, EffectElement, and IntroducedElement. Although the diagram suggests this is mainly happening at a structural level (the boxes symbolize element instances), note that the only way to discover the incompleteness of a situation description is when influence resolution (based on the complete set of currently known influences) yields contradictory results. Thus, the quantity value assignments play an important role in completing the situation description.

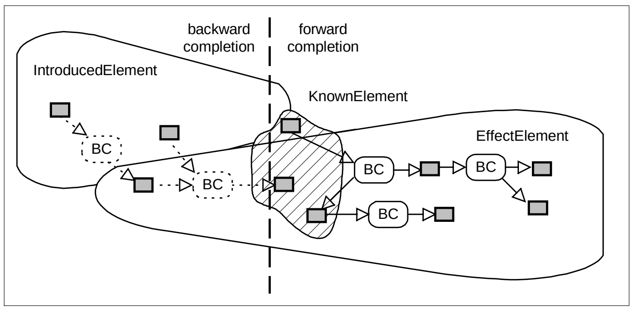

Figure 6 Completion of situation description: "forward" vs. "backward" and extensions of predicates

The diagram is intended as an intuitive illustration. Causation mediated by behavior constituents is indicated by rounded rectangles with the label "BC", arrows into them represent conditions and arrows out of behavior constituents stand for (structural) effects.

Unfortunately, obtaining a valid forward completion is not at all straightforward, due to the non-monotonicity inherent in the semantics: In collecting a set of structural elements as being part of the forward completion, one has to apply closure assumptions in order to compute values for the associated quantities (in the case of influence combinations) - but these values are in turn necessary to determine the occurrence of behavior constituents (via their quantity effects), which once again have an impact on the existence of other structural elements within the forward completion, thus challenging the closure assumption.

The key idea is to compute a superset of all initial forward completions by neglecting part of the occurrence conditions, namely the quantity conditions. Thus, we will collect all behavior constituents that can *potentially* occur, as far as their structural conditions are concerned. At the same time we have to collect their (potential) structural effects - and the behavior constituents that could result from these and so on. As a main advantage, this superset, we will call it the *extended forward space*, can usually be constructed in a strictly monotonic fashion from the initial situation description specified by the user. Figure 7 depicts the extended forward space:

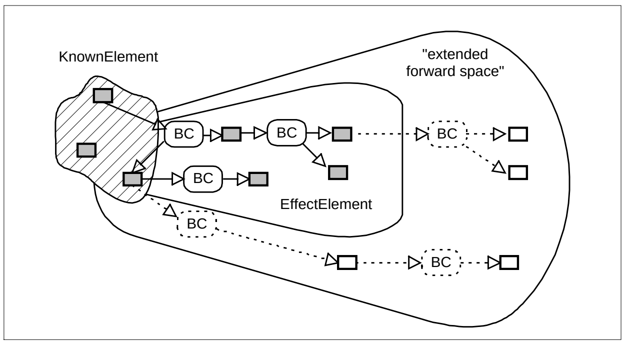

Figure 7 The "extended forward space" as a superset of the forward completion

Since we may include behavior constituents (and their effects) that might never actually occur - because their quantity conditions are never met - we have to create a conditional structure to prevent these effects. This separation of structural and quantity conditions corresponds to the distinction between "instantiation" and "activation" of processes in QPE.

The resulting set of behavior constituents and the associated conditional structure is then checked for completeness, or more precisely: it is checked whether there is a solution contained in the extended forward space. This is achieved on the basis of standard constraint solving and assumption tracking.

The Extended Forward Space - Instantiation Rules

For discussing the chosen algorithmic approach, we will employ a semi-formal description in the following.

We define *potential* behavior constituents with *instantiation and activation rules*. The extended forward space is constituted by structural elements that are *potentially* part of the forward completion, specified by

PotentialElement(e)
PotentialBC(pbc)

As the central *instantiation rules* for potential behavior constituents, we state:

(IR_1) PotentialSC $\rightarrow$ $\exists$pbc PotentialBC(pbc)

(IR_2) PotentialBC(pbc) $\rightarrow$ PotentialSE

The first instantiation rule (IR_1) states that a potential behavior constituent is present if and only if the structural conditions are *potentially fulfilled*, i. e. fulfilled in terms of potential elements being present. According to the second instantiation rule (IR_2), the potential structural effects are created together with the potential behavior constituent.

The starting point for the construction of the extended forward space is given by a "copy" of the initial situation description. The following important theorem provides the first part of the evidence that the presented reasoning approach is equivalent to the original semantics:

Theorem 1

If the backward completion is empty, i.e. no additional elements have been introduced, there exists a subset of the extended forward space that is structurally equivalent to the forward completion of the situation description.

Activation Rules

A potential behavior constituent requires two additional conditions to be fulfilled to actually occur, i. e. to become active: Obviously, the quantity conditions have been neglected, but also the structural conditions so far merely reference *potential* structural elements.

We employ a guard predicate, exists, for all structural elements, to distinguish between their potential and their actual existence. A second guard predicate, active, determines whether the potential behavior constituent actually *occurs* - and imposes its defined effects. See the following rules for the interdependencies:

(AR_1) PotentialBC(pbc) $\wedge$ SC' $\wedge$ QC' $\rightarrow$ active(pbc)

(AR_2) PotentialBC(pbc) $\wedge$ active(pbc) $\rightarrow$ SE' $\wedge$ QE'

The predicates for quantity conditions and effects, QC' and QE', accommodate quantities associated with potential objects. The first activation rule (AR_1) determines the activity of the potential behavior constituent from the fulfillment of the structural and quantity conditions. (AR_2) establishes the structural and quantity effects of active behavior constituents.

Again, there is a starting point for determining existence and activity: The only potential structural elements in the extended forward space known to exist are the ones from the initial situation description.

We can add the requirement that for an element to exist, it has to be either part of the initial situation description or the structural effect of an active behavior constituent. Using this stricter rule and for the case that there no loops in the creation structure, i. e. the structural effects of behavior constituents do not lead to the creation of their own structural conditions, we can prove the second important theorem for establishing the correctness of the computational approach:

Theorem 2

If there are no loops in the creation structure, then potential behavior constituents are active and potential structural elements exist, if *and only if* they are part of the structurally equivalent subset defined in theorem 1 (i. e. they have actual counterparts).

Proofs of both theorems can be found in [Heller 2001].

Model Fragments in the Extended Forward Space

We use boolean guard variables corresponding to the guard predicates for the "actual" existence of elements and for the activity of behavior constituents. Together with the rules discussed above, a behavior constituent is compiled into a network of constraints and influences, as illustrated in figure 8 for the process of iron redissolving.

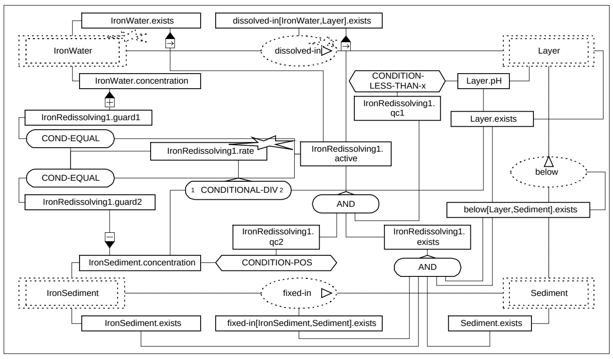

Figure 8 The complete influence diagram fragment for an instance of iron redissolving

After influence resolution in the composed model, we will obtain a constraint network that computes the active and existing elements of the extended forward space and, of course, values of variables, as well.

Influence Resolution and Local Closed-World Assumptions

The core problem is the determination of the *completeness* of the model. Our approach relies on assumption-based reasoning, and by making the closure of the model an explicit assumption, it can be retracted as part of the diagnostic process.

The closed-world assumption is the basis for each individual resolution of influences on a common target quantity. For this, we require that all influences are known. When taking the set of all influences in the extended forward space, we can define what we call the *local closed-world assumption* for a given quantity - that there is no additional influence besides the ones already collected.

For structural effects, the activation rule relies on the model closure, as well. Namely, the creation of a structural element (encoded as assigning a value of true to the associated exists variable) requires an active potential behavior constituent featuring the element in its structural effects. Again, the set of potential behavior constituents collected in the extended forward space has to be determined as complete. Analogously, we can define a local closed-world assumption for a potential structural element, stating that there is no additional behavior constituent supporting it.

Searching for Backward Completions

Backward completion will start with a specific local closed-world assumption. For local CWAs from quantities, the first step is to identify the object the quantity is associated with and - based on object type and quantity role - to select behavior constituent types that contain an

influence on such a quantity in their quantity effects. For local CWAs from structural elements, the respective object type will have to be found in the structural effects. In each case, a partial mapping for a selected behavior constituent type will be constructed that will bind the respective object to the required role - and the structural completion algorithm will be started again, but this time in "backward mode".
For the current G$^+$DE prototype, a "brute force" algorithm for the overall backward search has been implemented, which finds all supports for all creatable elements from intermediate search results (not shown here). Currently, this is the only option besides the interactive approach, which relies on the user to define the search path. Certainly, future versions will incorporate more sophisticated search modules with focusing strategies or application-specific heuristics.

Minimality of Backward Completions

The crucial point is that we never introduce elements without need. And the algorithmic elements for backward search proceed in a way that guarantees minimality of the retrieved configurations ("structural augmentations") - locally. But be aware that the global requirement for minimality might have to be checked for the complete constructed augmentation.
While we can easily establish a final check for the minimality of augmentation candidates retrieved, it is much more difficult to find valid focusing principles for the searching algorithm.

System Architecture of G$^+$DE

Figure 9 depicts the architecture of G$^+$DE at an abstract level. A *Domain Theory Editor* (upper left corner) enables the user to design and modify the domain theory with a graphical user interface. A *Scenario Editor* allows the creation of a situation description, which is decomposed into the system structure and quantity specifications. Quan-ity specifications can contain user-defined assumptions.
Structure Completion constructs the extended forward space from the domain theory and the system structure. In the course of completion, the system structure is extended. As a result, a set of behavior constituent instances is passed to *Constraint Net Generation*, which handles the encoding of all conditions and effects into the conditional structure, as well as influence resolution and optional transfor-ations. The resulting constraint net is basically an encoding of the extended forward space including the complete conditional structure for reasoning about existence and activation.
A GDE module then uses this constraint net ("SD") plus the quantity specifications ("OBS") and proceeds with prediction, conflict detection and diagnostic candidate generation. Note that the user-defined assumptions asso-iated with certain quantity specifications have to be taken into account here. This achieves the consistency check. Candidates that contain closed-world assumptions are handled in the *Model Revision* module. The fundamental backward search algorithm relies on the Structure Com-letion once more, but the backward arrow indicates also the generate-and-test loop for the completed structure.

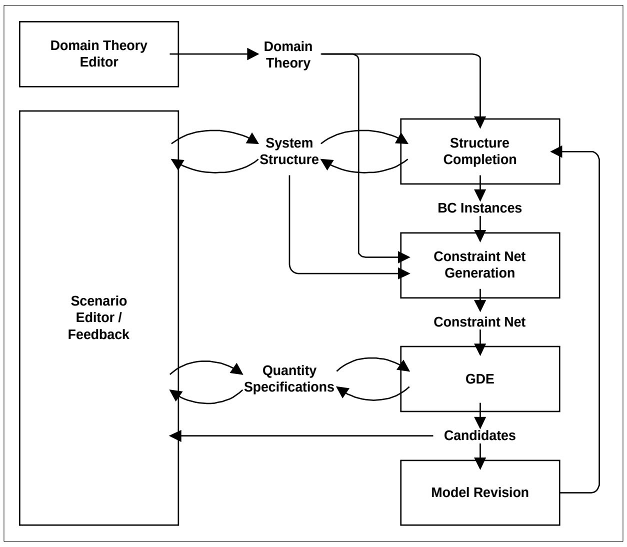

Figure 9 Architecture of G$^+$DE (abstract view)

The Scenario Editor is also intended for all user feedback by displaying the augmented structure, the computed quantity values, and diagnostic candidates. It can also be used for interactively selecting search paths for model revision.

Related Work and Discussion

GDE, GDE$^+$, and Sherlock

It turns out that most classical approaches can be seen as special cases of the generalized theory presented here. However, they excel in smart instantiation of behavior model fragments. See [Heller 2001] for a discussion of the simplifications leading to the classical approaches.

PDE

The work of Collins ([Collins 1993]) is the one that comes closest to the presented approach. Models built within the original Qualitative Process Theory ([Forbus 1984]) are used as input to the Process-based Diagnostic Engine, PDE. Collins employs local closed-world assumptions in much the same way as we do. However, once diagnostic candidates containing closed-world assumptions are discovered, Collins proceeds in two steps by first hypothe-izing a "process structure", i. e. a set of (active) process instances - and then searching abductively for an underlying object structure.
But apparently, PDE is not built for handling general structural effects - or "contingent objects" in QPT terminology. Collins neither discusses the complex issues

involved, nor are they featured in any of his examples. A review of the sections on model composition and model revision reveals the extent of the simplification thus achieved.

Diagnosis of Structural Faults

There have been various attempts to overcome the dependency of component-oriented diagnosis on a fixed structure and to cover "structural faults". [Böttcher 1995] is a particularly promising approach, in which a potential interaction is represented as a special kind of model fragment, a "hidden interaction". The definition of "hidden interaction models" is surprisingly close to what we would call a behavior constituent type definition and, thus, a reconstruction of the approach within our framework is simple.
However, the correct behavior is still restricted to being modeled as a one-to-one mapping of components to model fragments. The process-oriented modeling scheme presented here goes a step further. See [Heller 2001] for examples of diagnosing structural faults with G^+DE.

Limitations

One of the main limitations of the presented theory is the *state-based approach*: all structural statements as well as all observations and predictions refer to the same instant in time, which largely facilitates consistency checking. Note that the enhancement of a model with derivatives or deviation models partly compensates the need for temporal representations. See also the fundamental results of [Struss 1997] for the diagnosis of dynamic systems without with state-based approaches.
A concern arising with the current prototype is the impact of the model composition on *efficiency*. For complex domain theories, potentially a large number of behavior constituents are collected, while many of them might never be activated. For certain classes of domain theories, of course the composition step could be extended to take part of the quantity conditions into account, so that the creation of certainly inactive instances would be avoided.

Advantages of the Approach

It is worth mentioning that our work builds on the *rigorous formal foundations* of consistency-based diagnosis and extends and generalizes them.
Even more so, G^+DE is more specific with respect to the structure of system models allowed. It thus requires a certain stringency in structure-to-behavior reasoning. In effect, we preserve and enforce the *approach from "first principles"* the research field is based on.
Note that the closure of the system model is a necessary requirement for most prediction and diagnosis tasks. However, usually the system boundary is implicit and will never be challenged. This is one of the areas, where our generalized theory provides a clear and formal alternative in *making the closure assumption explicit*.
As evidence of the *generality and broad applicability* of the presented theory and the implemented prototype see [Heller 2001]. There, not only the example from this paper but also a number of extended application examples from the areas of ecology, medicine and electrical circuits are solved, highlighting the powerful search for additional elements, the treatment of feedback loops and degradation faults, as well as diagnosis of structural faults and the exploitation of structural configurations.

References

[Böttcher 1995] C. Böttcher: *No Faults in Structure? - How to Diagnose Hidden Interactions.* In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, pp. 1728-1734, Morgan Kaufmann Publishers, 1995.

[Collins 1993] J. Collins: *Process-based Diagnosis: An Approach to Understanding Novel Failures.* Doctoral thesis, Institute for the Learning Sciences, Northwestern University, 1993.

[de Kleer/Williams 1987] J. de Kleer and B. Williams: *Diagnosing Multiple Faults.* Artificial Intelligence, 32(1), 1987.

[de Kleer/Williams 1989] J. de Kleer and B. Williams: *Diagnosis with Behavioral Modes.* In: Proceedings of the 11th International Joint Conference on Artificial Intelligence, 1989.

[Dressler/Struss 1994] O. Dressler, P. Struss: *Model-based Diagnosis with the Default-based Diagnosis Engine: Effective Control Strategies that Work in Practice.* In: Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94), John Wiley, 1994.

[Heller 2001] Heller, U., *Process-oriented Consistency-based Diagnosis - Theory, Implementation and Applications.* Doctoral thesis, Department of Computer Science, Technical University of Munich, 2001.

[Reiter 1980] R. Reiter: *A Logic for Default Reasoning*. In: Artificial Intelligence, 13:81-132, 1980.

[Reiter 1987] R. Reiter: *A Theory of Diagnosis from First Principles.* Artificial Intelligence, 32(1):57-96, 1987.

[Struss 1997] P. Struss: *Fundamentals of Model-Based Diagnosis of Dynamic Systems.* In: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, pp. 480-485, 1997.

[Struss/Dressler 1989] P. Struss and O. Dressler: *'Physical Negation' - Integrating Fault Models into the General Diagnostic Engine.* In: Proceedings of the 11th International Joint Conference on Artificial Intelligence, 1989.

[Struss/Heller 1998] P. Struss and U. Heller: *Process-oriented Modeling and Diagnosis - Revising and Extending the Theory of Diagnosis from First Principles.* In: 9th International Workshop on Principles of Diagnosis (DX-98), Cape Cod, MA, USA, pp. 110-117, 1998.