Derived Transaction Termination Dependencies: An Algorithm for Computing Transitivity Rules

Kerstin Schwarz   Can Türker   Gunter Saake

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
E-mail: {schwarz|tuerker|saake}@iti.cs.uni-magdeburg.de

February 1998

Abstract

Complex applications consist of a large set of transactions which are interrelated. There are different kinds of dependencies among transactions of a complex application, e.g. termination dependencies which are constraints on the occurrence of significant transaction events. Often, it is very difficult for the application/transaction designer to get a grasp of the transitive relationships among the transactions of a complex application. In this paper, we introduce the notion of *transaction closure* as a generalization of nested transactions. A transaction closure comprises all transactions which are transitively initiated by one (root) transaction. The relationships among transactions of a transaction closure are specified by different kinds of dependencies. We identify transaction termination dependencies which determine reasonable termination event combinations of two related transactions. These basic termination dependencies are further refined considering the aspect of compensatable transactions. In particular, we consider the transitivity property for all kinds of transaction termination dependencies discussed in this paper and present a set of rules for reasoning about the transitivity of these dependencies. Thus, we are able to conclude how two arbitrary transactions are transitively interrelated. This issue is essential for understanding the entire semantics of a complex application.

Keywords: transaction closure, termination dependencies, compensation, transitive dependencies.

Contents

1 Introduction **1**

2 The Notion of Transaction Closures **3**

3 Termination Dependencies in Transaction Closures **5**

3.1 Basic Termination Dependencies 5

3.2 Influence of Transaction Compensation 8

4 Transitivity of Termination Dependencies **13**

4.1 Foundations of Derived Dependencies 13

4.2 Computing Valid Dependency Combinations 16

4.3 Exemplary Application of the Algorithm 22

5 Application of the Transitivity Rules **25**

6 Conclusions and Outlook **29**

Bibliography **31**

A Transitivity Rules of Termination Dependencies **35**

Chapter 1

Introduction

The dynamics of complex applications can be modeled by collections of related transactions. Examples for such complex applications are business processes, CSCW applications or design transactions. In such applications there are different kinds of dependencies among the related transactions, e.g. termination dependencies which are constraints on the occurrence of significant transaction events. Often, it is very difficult for the application/transaction designer to get a grasp of the transitive relationships among the transactions of a complex application.

The concept of *transaction closure* [STS98] as a generalization of the well-known concept of nested transactions [Mos85] together with a set of transaction termination dependencies provide a common framework for modeling advanced, complex transactions/activities. A transaction closure comprises of a set of transactions which are transitively initiated by the same (root) transaction. In contrast to nested transactions, a child transaction in a transaction closure may survive the termination of its parent transaction — a case which is needed for example in models for long-during activities [DHL91, RKT+95], workflows [GHS95, KR96], or in active databases [HLM88, BÖH+92].

Termination dependencies play a central role in transaction closures. A termination dependency is a constraint on the possible combinations (and orders) of the termination events (commit/abort) of two related transactions. In transaction closures, the framework for termination dependencies has to be extended in comparison to nested transactions. Besides the termination combinations, one has to consider, for instance, the effects of compensating transactions on termination dependencies. Whereas the termination condition for direct child transactions are explicitly specified, the conditions for transitively related transactions have to be computed based on the direct dependencies.

In this paper, we present a framework for handling termination conditions for transitive ancestor relations in transaction closures. Here, we also consider the aspect of transaction compensation. The formalization of transaction closures and dependencies is based on the ACTA framework presented in [CR90, CR91, CR94]. The result is a practical algorithm for computing derived dependencies for transitively related transactions. Furthermore, the rules obtained by this algorithm together with the concept of transaction closure and the basic dependencies provide the basis for a transaction design and analyzing tool. Such a tool can help to understand the entire semantics of a complex application and thus it may support the design of better and more efficient applications.

The paper is organized as follows: In Section 2, we introduce the notion of transaction closure. Thereafter, in Section 3, we discuss transaction termination dependencies which deal with valid combinations of termination events (commit and abort) of related transactions of a transaction closure. Furthermore, the effect of transaction compensation on termination dependencies is analyzed. In Section 4, we consider the transitivity of the dependencies introduced and develop an algorithm for deriving all valid combinations of dependencies for a given transitive dependency. The application of transaction closures, especially the derivation of transitive dependencies in transaction closures, is shown by an example in Section 5. Finally, the paper is concluded by an outlook on future work.

Chapter 2

The Notion of Transaction Closures

Traditionally, advanced transaction models base on the idea of nesting transactions. A nested transaction [Mos85] is a transaction consisting of several transactions called *child transactions*. These child transactions are initiated within a transaction. The initiating transaction is called *parent*. A child transaction itself may consist of other child transactions, i.e., a nested transaction forms a transaction tree. However, activity models [DHL91] or workflow models [KR96] require a more general framework for describing sets of related transactions. The same holds for execution models of active database systems [HLM88]. We introduce the notion of *transaction closure* as a generalized transaction structure allowing greater independence between constituting transactions.

A transaction closure consists of a set of transactions which are transitively initiated by the same (root) transaction, i.e., there are parent-child relationships between transaction pairs of a transaction closure. The transactions of a transaction closure are further interrelated by the dependencies we discuss in the following chapters. For example, a child transaction in a transaction closure may leave the scope of its parent transaction because of a special termination dependency between them. By using transaction closures (in connection with the various dependency types) we have a uniform framework for describing traditional as well as advanced transaction models, particularly activity and workflow models. For the definition of the notion of a transaction closure we first define some basic notions.

Definition 2.0.1 *The following self-explanatory functions and predicates describe general relationships between a transaction and its initiator:*

$$\begin{aligned}
parent(t_i, t_j) &:= (\, t_i \text{ is parent of } t_j \,) \\
root(t_j) &:= (\, t_j \text{ has no parent} \,) \\
ancestor(t_i, t_j) &:= (\, parent(t_i, t_j) \vee (\exists t_k : ancestor(t_i, t_k) \wedge parent(t_k, t_j)) \,)
\end{aligned}$$

The relationship between the initiation of two transactions can be expressed by a *begin dependency*. For our definitions we use the ACTA formalism [CR90, CR91, CR94, RC97], including the fundamental dependency definitions such as the *abort dependency* and *exclusive dependency* and the notion of a history of transactions [BHG87, GR93] (in the following formulas denoted as H). A single arrow ($\rightarrow$) denotes temporal sequence.

Begin Dependency $(t_i\, \mathcal{BD}\, t_j)$**.** A transaction t_i can be initiated only if transaction t_j is already initiated:

$$(begin_{t_i} \in H) \Rightarrow (begin_{t_j} \rightarrow begin_{t_i})$$

Definition 2.0.2 (Transaction Closures) *Suppose tc denotes a set of transactions of a transaction closure and let t_i and t_j be two transactions of this closure:*

- *Each transaction closure has* exactly one[1] *root transaction:*

$$\exists!\, t_i \in tc : root(t_i)$$

- *Each non-root transaction has* exactly one *parent transaction:*

$$\forall t_j \in tc : \neg root(t_j) \Rightarrow (\exists!\, t_i \in tc : parent(t_i, t_j))$$

- *Each transaction closure is acyclic:*

$$\nexists t_i \in tc : ancestor(t_i, t_i)$$

- *The initiation of a transaction must follow the initiation of the parent:*

$$\forall t_j \in tc : \neg root(t_j) \Rightarrow (\exists t_i \in tc : parent(t_i, t_j) \wedge (t_j\, \mathcal{BD}\, t_i))$$

In the following chapters, we discuss different termination dependencies between transactions of a transaction closures. These dependencies have to be defined (at least) between a transaction and its parent. As we will see later, these dependencies have transitively effects on the other transactions of the closure.

[1] The symbol $\exists!$ stands for "it exists exactly one".

Chapter 3

Termination Dependencies in Transaction Closures

In addition to the begin dependency, there are several dependencies among transactions of a transaction closure. In this paper we concentrate on *termination dependencies* which are constraints on the occurrence of the significant termination events and show the derivation of transitive termination dependencies.

3.1 Basic Termination Dependencies

Investigating constraints on the occurrence of the significant termination events commit and abort leads to different termination dependencies. In case of two transactions t_i and t_j there are four possible combinations of termination events:

- **(1)** both transactions abort (a_{t_i}, a_{t_j})
- **(2/3)** one transaction commits whereas the other one aborts $(a_{t_i}, c_{t_j})/(c_{t_i}, a_{t_j})$
- **(4)** both transactions commit (c_{t_i}, c_{t_j})

These termination event combinations may be valid in any order (denoted by $\surd$) or are not valid (denoted by —). Obviously, there are at most sixteen possible dependencies which are formed by different values for the termination event combinations, e.g. all combinations are valid (see the last column in Table 3.1). Under the assumption that a transaction may be forced to abort but not to commit, the number of possibly reasonable termination dependencies is reduced to at most eight. This is due to the fact that the abortion of both transactions is considered as valid event combination for all dependencies. We identify five of the eight dependencies as reasonable according to real-world application semantics. These dependencies are summarized in Table 3.1.

The remaining three dependencies are inapplicable according to real-world application semantics. These are the cases (1) in which only the abortion of both transactions is

t_i	t_j	$vital\_dep(t_i, t_j)$	$vital(t_i, t_j)$	$dep(t_i, t_j)$	$exc(t_i, t_j)$	$indep(t_i, t_j)$
a_{t_i}	a_{t_j}	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
a_{t_i}	c_{t_j}	—	—	$\surd$	$\surd$	$\surd$
c_{t_i}	a_{t_j}	—	$\surd$	—	$\surd$	$\surd$
c_{t_i}	c_{t_j}	$\surd$	$\surd$	$\surd$	—	$\surd$

Table 3.1: Termination Dependencies between two Transactions t_i and t_j

valid, and (2 and 3) in which the abortion of one transaction and the commitment of the other transaction are valid in addition to the first case. The dependency where only the abortion of both transaction is valid makes no sense because in this case no transaction is allowed to commit. In other words, if transactions are only allowed to abort, they do not need to be executed. The last two dependencies are also not reasonable because in these cases always one of the related two transactions is not allowed to commit, independently of the termination event of the other transaction.

After having discussed the non-reasonable termination event combinations, we will now define the termination dependencies which are the basis for the reasonable termination events combinations.

Abort Dependency $(t_i \,\mathcal{AD}\, t_j)$**.** If transaction t_j aborts, then transaction t_i has to abort, too:

$$(abort_{t_j} \in H) \Rightarrow (abort_{t_i} \in H)$$

Exclusive Dependency $(t_i \,\mathcal{ED}\, t_j)$**.** If transaction t_j commits, then transaction t_i has to abort:

$$(commit_{t_j} \in H) \Rightarrow (abort_{t_i} \in H)$$

A dependency between two transactions t_i and t_j which requires that both transactions either abort or commit together is called *vital-dependent*.

Definition 3.1.1 (Vital-Dependent) *Two different transactions t_i and t_j are* vital-dependent *on each other if and only if both transactions are* abort dependent *on each other:*

$$vital\_dep(t_i, t_j) \quad :\Leftrightarrow \quad (t_i \,\mathcal{AD}\, t_j) \wedge (t_j \,\mathcal{AD}\, t_i)$$

The vital-dependent dependency is (as the name suggests) a combination of the dependencies *vital* and *dependent* which are defined below.

Definition 3.1.2 (Vital) *A transaction t_i is* vital *for another transaction t_j if and only if t_j is* abort dependent *on t_i:*

$$vital(t_i, t_j) \quad :\Leftrightarrow \quad (t_j \,\mathcal{AD}\, t_i)$$

Thus, the *vital* dependency between two transactions t_i and t_j concerns the case where the abortion of transaction t_i leads to the abortion of transaction t_j. In comparison to a vital transaction, a *dependent* transaction t_j has to abort if transaction t_i aborts. Thus, the termination event of transaction t_i is necessary for an acceptable outcome according to the semantics of the application. In contrast, the results of transaction t_j are not essential for the application.

Definition 3.1.3 (Dependent) *A transaction t_i is* dependent *on another transaction t_j if and only if t_i is* abort dependent *on t_j:*

$$dep(t_i, t_j) \quad :\Leftrightarrow \quad (t_i \, \mathcal{AD} \, t_j)$$

From the Definitions 3.1.2 and 3.1.3 directly follows that either *vital* or *dependent* is not required because these dependencies can be substituted by each other ($vital(t_i, t_j) \equiv dep(t_j, t_i)$). This observation is true if we do not consider combinations of termination dependencies with other kinds of dependencies, e.g. the so-called object visibility constraints which determine when results of a transaction are made visible to other transactions. The combination of termination dependencies with object visibility dependencies leads to different results for $vital(t_i, t_j)$ and $dep(t_j, t_i)$ due to the fact that a transaction which make its effects only visible to its parent requires that its termination precedes the commitment of the parent. In other words, the parent transaction has to wait for the termination of such a child transaction to get the child's results. Since the dependent and vital termination dependencies cannot be substituted in this case, we explicitly distinguish these two termination dependencies.

A completely different dependency is the *exclusive* dependency. Here, only one of the transactions is allowed to finish successfully. This dependency may be useful, for instance, in real-time applications where two alternative transactions are executed in parallel and the results of the transaction which finishes at first are accepted and the results of the other one is aborted after that.

Definition 3.1.4 (Exclusive) *Two different transactions t_i and t_j are* exclusive *for each other if and only if both transactions are* exclusive dependent *on each other:*

$$exc(t_i, t_j) \quad :\Leftrightarrow \quad (t_i \, \mathcal{ED} \, t_j) \wedge (t_j \, \mathcal{ED} \, t_i)$$

Our fifth dependency concerns the case where each combination of transaction termination events is valid. Therefore, the involved transactions are denoted as *independent*. Concurrently executed transactions without any constraints on the occurrence of the termination events are independent. A possible scenario for independent transactions is the situation in which two different sources offer the same service and two (or more) transactions are executed in parallel to get the required results.

Definition 3.1.5 (Independent) *Two different transactions t_i and t_j are* independent *if all termination event combinations of t_i and t_j are allowed. This case is denoted as $indep(t_i, t_j)$.*

After having a closer look on the definitions above, we can conclude that $vital(t_i, t_j)$ is equivalent to $dep(t_j, t_i)$. In other words, transaction t_i is vital for t_j if and only if t_j is dependent on t_i and vice versa. In contrast, the other dependencies are symmetrical (by definition).

Theorem 3.1.6 *The following relationships hold for dependencies between two transactions t_i and t_j:*

$$\begin{aligned} vital\_dep(t_i, t_j) &\equiv vital(t_i, t_j) \wedge dep(t_i, t_j) && (3.1) \\ vital\_dep(t_i, t_j) &\equiv vital\_dep(t_j, t_i) && (3.2) \\ vital(t_i, t_j) &\equiv dep(t_j, t_i) && (3.3) \\ exc(t_i, t_j) &\equiv exc(t_j, t_i) && (3.4) \\ indep(t_i, t_j) &\equiv indep(t_j, t_i) && (3.5) \end{aligned}$$

3.2 Influence of Transaction Compensation on Termination Dependencies

In this section, we will see that the termination dependencies can be weakened when transaction compensation is supported. Transaction compensation is the ability to semantically undo the effects of a transaction t_i by a *compensating* transaction $comp_{t_i}$ in order to achieve semantic atomicity [Gar83]. If there exists such a compensating transaction $comp_{t_i}$ for a transaction t_i, t_i is denoted as *compensatable*.

The basic termination dependencies defined in Section 3 base on the abort dependency and the exclusive dependency, respectively. If transaction compensation is considered, these dependencies are weakened to the *weak-abort dependency*[1] and the *weak-exclusive dependency*, respectively.

Weak-Abort Dependency $(t_i\, \mathcal{WAD}\, t_j)$**.** If transaction t_i commits and transaction t_j aborts, then the commitment of t_i precedes the abortion of t_j and the compensating transaction $comp_{t_i}$ of t_i has to commit:

$$((abort_{t_j} \in H) \wedge (commit_{t_i} \in H)) \Rightarrow (commit_{t_i} \rightarrow abort_{t_j} \wedge (commit_{comp_{t_i}} \in H))$$

In contrast to an abort dependent transaction, a weak-abort dependent transaction can commit without waiting for the commit of the other transaction. In case the other transaction aborts, the "committed" effects of the weak-abort dependent transaction are semantically undone by executing a compensating transaction.

[1]Please note that our definition of the weak-abort dependency differs from the weak-abort dependency $(t_i\, \mathcal{WD}\, t_j)$ defined in [CR94], because we here integrate the aspect of compensation.

Weak-Exclusive Dependency $(t_i\, \mathcal{WED}\, t_j)$**.** If both transactions t_i and t_j commit, then the compensating transaction $comp_{t_i}$ of t_i has to commit, too:

$$((commit_{t_j} \in H) \wedge (commit_{t_i} \in H)) \Rightarrow (commit_{comp_{t_i}} \in H)$$

In comparison to the exclusive dependency, the weak-exclusive dependency $(t_i\, \mathcal{WED}\, t_j)$ only demands that the compensating transactions of t_i must commit in case both transactions t_i and t_j are committed. Thus, the case where both transactions can commit is allowed under the restriction that transaction t_i is compensatable.

By using the weakened versions of the abort and the exclusive dependency, we now redefine the termination dependencies under the consideration that the related transactions are compensatable (at least one of them). For the symmetrical dependency $vital\_dep(t_i, t_j)$ we have to distinguish three cases: (1) $vital\_dep_{◀}(t_i, t_j)$ if only t_i is compensatable, (2) $vital\_dep_{▶}(t_i, t_j)$ if only t_j is compensatable, and (3) $vital\_dep_{⧫}(t_i, t_j)$ if both t_i and t_j are compensatable.

Definition 3.2.1 (Vital-Dependent$_{◀}$) *Two transactions t_i and t_j are* vital-dependent$_{◀}$ *on each other if and only if t_i is* weak-abort dependent *on t_j and t_j is* abort dependent *on t_i:*

$$vital\_dep_{◀}(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{WAD}\, t_j) \wedge (t_j\, \mathcal{AD}\, t_i)$$

Definition 3.2.2 (Vital-Dependent$_{▶}$) *Two transactions t_i and t_j are* vital-dependent$_{▶}$ *on each other if and only if t_i is* abort dependent *on t_j and t_j is* weak-abort dependent *on t_i:*

$$vital\_dep_{▶}(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{AD}\, t_j) \wedge (t_j\, \mathcal{WAD}\, t_i)$$

Definition 3.2.3 (Vital-Dependent$_{⧫}$) *Two transactions t_i and t_j are* vital-dependent$_{⧫}$ *on each other if and only if both transactions are* weak-abort dependent *on each other:*

$$vital\_dep_{⧫}(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{WAD}\, t_j) \wedge (t_j\, \mathcal{WAD}\, t_i)$$

For the asymmetric relationships vital and dependent, only one direction has to be handled.

Definition 3.2.4 (Vital$_{▶}$) *A transaction t_i is* vital$_{▶}$ *for another transaction t_j if and only if t_j is* weak-abort dependent *on t_i:*

$$vital_{▶}(t_i, t_j) \quad :\Leftrightarrow \quad (t_j\, \mathcal{WAD}\, t_i)$$

In other words, if $vital_{▶}(t_i, t_j)$ holds, then transaction t_i is vital for the compensatable *transaction t_j.*

Definition 3.2.5 (Dependent$_◀$) *A transaction t_j is* dependent$_◀$ *on another transaction t_i if and only if t_i is* weak-abort dependent *on t_j:*

$$dep_◀(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{WAD}\, t_j)$$

In other words, if $dep_◀(t_i, t_j)$ holds, then the compensatable *transaction t_i is dependent on t_j.*

As in the vital-dependent case, we have also to distinguish three cases for the exclusive dependency.

Definition 3.2.6 (Exclusive$_◀$) *Two transactions t_i and t_j are* exclusive$_◀$ *if and only if t_i is* weak-exclusive dependent *on t_j:*

$$exc_◀(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{WED}\, t_j)$$

Definition 3.2.7 (Exclusive$_▶$) *Two transactions t_i and t_j are* exclusive$_▶$ *if and only if t_j is* weak-exclusive dependent *on t_i:*

$$exc_▶(t_i, t_j) \quad :\Leftrightarrow \quad (t_j\, \mathcal{WED}\, t_i)$$

Definition 3.2.8 (Exclusive$_\blacklozenge$) *Two transactions t_i and t_j are* exclusive$_\blacklozenge$ *for each other if and only if either t_i is* weak-exclusive dependent *on t_j or t_j is* weak-exclusive dependent *on t_i:*

$$exc_\blacklozenge(t_i, t_j) \quad :\Leftrightarrow \quad (t_i\, \mathcal{WED}\, t_j) \vee (t_j\, \mathcal{WED}\, t_i)$$

t_i	t_j	$vital\_dep_\blacklozenge(t_i, t_j)$	$vital\_dep_◀(t_i, t_j)$	$vital\_dep_▶(t_i, t_j)$	$vital_▶(t_i, t_j)$
a_{t_i}	a_{t_j}	$\surd$	$\surd$	$\surd$	$\surd$
a_{t_i}	c_{t_j}	$c_{t_j} \rightarrow a_{t_i} \wedge c_{comp_{t_j}}$	—	$c_{t_j} \rightarrow a_{t_i} \wedge c_{comp_{t_j}}$	$c_{t_j} \rightarrow a_{t_i} \wedge c_{comp_{t_j}}$
c_{t_i}	a_{t_j}	$c_{t_i} \rightarrow a_{t_j} \wedge c_{comp_{t_i}}$	$c_{t_i} \rightarrow a_{t_j} \wedge c_{comp_{t_i}}$	—	$\surd$
c_{t_i}	c_{t_j}	$\surd$	$\surd$	$\surd$	$\surd$

t_i	t_j	$exc_\blacklozenge(t_i, t_j)$	$exc_◀(t_i, t_j)$	$exc_▶(t_i, t_j)$	$dep_◀(t_i, t_j)$
a_{t_i}	a_{t_j}	$\surd$	$\surd$	$\surd$	$\surd$
a_{t_i}	c_{t_j}	$\surd$	$\surd$	$\surd$	$\surd$
c_{t_i}	a_{t_j}	$\surd$	$\surd$	$\surd$	$c_{t_i} \rightarrow a_{t_j} \wedge c_{comp_{t_i}}$
c_{t_i}	c_{t_j}	$c_{comp_{t_i}} \vee c_{comp_{t_j}}$	$c_{comp_{t_i}}$	$c_{comp_{t_j}}$	$\surd$

Table 3.2: Relaxed Termination Dependencies

Considering transaction compensation the basic termination dependencies are relaxed. All disallowed fields (except the two fields in $vital\_dep_◀(t_i, t_j)$ and $vital\_dep_▶(t_i, t_j)$, respectively) are replaced by a constraint on the termination order of the related transactions or by the requirement that there must be a committed compensating transaction (compare Table 3.1 and Table 3.2). These effects are due to the usage of the weak-abort and weak-exclusive dependencies instead of the basic abort and exclusive dependencies. From the Definitions 3.2.3 to 3.2.8, we can derive the following theorem.

Theorem 3.2.9 *The following relationships hold for dependencies between two transactions t_i and t_j:*

$$\begin{aligned}
vital_dep_{\blacklozenge}(t_i,t_j) &\equiv vital_{\blacktriangleright}(t_i,t_j) \wedge dep_{\blacktriangleleft}(t_i,t_j) && (3.6)\\
vital_dep_{\blacktriangleleft}(t_i,t_j) &\equiv vital(t_i,t_j) \wedge dep_{\blacktriangleleft}(t_i,t_j) && (3.7)\\
vital_dep_{\blacktriangleright}(t_i,t_j) &\equiv vital_{\blacktriangleright}(t_i,t_j) \wedge dep(t_i,t_j) && (3.8)\\
vital_dep_{\blacklozenge}(t_i,t_j) &\equiv vital_dep_{\blacklozenge}(t_j,t_i) && (3.9)\\
vital_dep_{\blacktriangleleft}(t_i,t_j) &\equiv vital_dep_{\blacktriangleright}(t_j,t_i) && (3.10)\\
vital_{\blacktriangleright}(t_i,t_j) &\equiv dep_{\blacktriangleleft}(t_j,t_i) && (3.11)\\
exc_{\blacklozenge}(t_i,t_j) &\equiv exc_{\blacklozenge}(t_j,t_i) && (3.12)\\
exc_{\blacktriangleleft}(t_i,t_j) &\equiv exc_{\blacktriangleright}(t_j,t_i) && (3.13)
\end{aligned}$$

Up to now, we have discussed termination dependencies with and without compensation. The following section investigates the transitivity of the dependencies introduced so far. The transitivity is fundamental for deriving the relationship between two arbitrary transactions of a transaction closure. In particular, it is required to reason about the consequences of a transaction's abortion on the other transactions.

Chapter 4

Transitivity of Termination Dependencies

A termination dependency may have a transitive influence on other transactions of the transaction closure. Suppose, X is a termination dependency between the transactions t_i and t_k and Y a dependency between the transactions t_k and t_j, then the combination of these two dependencies leads to a transitive dependency Z between the transactions t_i and t_j; i.e., $X(t_i, t_k) \wedge Y(t_k, t_j) \Rightarrow Z(t_i, t_j)$ holds.

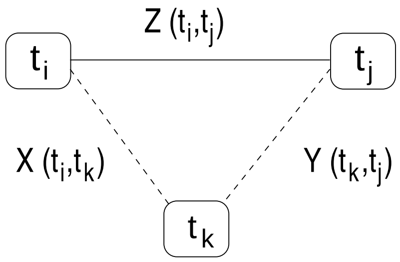

Figure 4.1: Example of a Transitive Dependency

Our approach is now to find all valid combinations of dependencies X and Y for a given transitive dependency Z (see Figure 4.1). For that, we analyze the basic relationships between the dependencies introduced so far. Based on the results how two dependencies can be combined, we present a method for deriving all valid combinations of dependencies X and Y for a transitive dependency Z. These valid dependency combinations result in rules which are the basis for a tool to assist the transaction designer in specifying transaction closures.

4.1 Foundations of Derived Dependencies

Before introducing the method to derive all valid combinations of dependencies for a given transitive dependency, we have to investigate the following basic formula in more detail:

$$X(t_i, t_k) \wedge Y(t_k, t_j) \Rightarrow Z(t_i, t_j) \tag{4.1}$$

Concerning the possible termination event combinations, we distinguish five possibilities for the application of the dependencies X, Y, and Z on a pair of transactions (denoted by the following symbols):

- $\ominus$ The combination of termination events is *not valid*. In this case both transactions are non-compensatable.
- $\oplus$ The termination event combination is *valid* (independent of the termination order and compensation properties) for a dependency.
- $\blacklozenge$ The combination of termination events is valid under the restriction that compensation of one of the two related transactions is possible. In this case both transactions have to be compensatable.
- $\blacktriangleleft$, $\blacktriangleright$ In comparison to $\blacklozenge$, in this case only one of the two related transactions is compensatable. The direction of the triangle indicates the transaction which has to be compensatable.

The usage of this symbols is illustrated in Table 4.1 (cf. Table 3.2).

t_i	t_j	$vital\_dep_{\blacklozenge}(t_i, t_j)$	$vital\_dep_{\blacktriangleleft}(t_i, t_j)$	$vital\_dep_{\blacktriangleright}(t_i, t_j)$	$vital_{\blacktriangleright}(t_i, t_j)$
a_{t_i}	a_{t_j}	$\oplus$	$\oplus$	$\oplus$	$\oplus$
a_{t_i}	c_{t_j}	$\blacktriangleright$	$\ominus$	$\blacktriangleright$	$\blacktriangleright$
c_{t_i}	a_{t_j}	$\blacktriangleleft$	$\blacktriangleleft$	$\ominus$	$\oplus$
c_{t_i}	c_{t_j}	$\oplus$	$\oplus$	$\oplus$	$\oplus$

t_i	t_j	$exc_{\blacklozenge}(t_i, t_j)$	$exc_{\blacktriangleleft}(t_i, t_j)$	$exc_{\blacktriangleright}(t_i, t_j)$	$dep_{\blacktriangleleft}(t_i, t_j)$
a_{t_i}	a_{t_j}	$\oplus$	$\oplus$	$\oplus$	$\oplus$
a_{t_i}	c_{t_j}	$\oplus$	$\oplus$	$\oplus$	$\oplus$
c_{t_i}	a_{t_j}	$\oplus$	$\oplus$	$\oplus$	$\blacktriangleleft$
c_{t_i}	c_{t_j}	$\blacklozenge$	$\blacktriangleleft$	$\blacktriangleright$	$\oplus$

Table 4.1: Graphically Representation of Termination Dependencies with Compensation

The transactions t_i, t_j, and t_k of the dependencies $X(t_i, t_k)$, $Y(t_k, t_j)$, and $Z(t_i, t_j)$ may either abort or commit. Thus, there are four possible termination event combinations for each dependency:

$$\begin{aligned} X(t_i, t_k) &\mapsto x_{(a,a)} \vee x_{(a,c)} \vee x_{(c,a)} \vee x_{(c,c)} \\ Y(t_k, t_j) &\mapsto y_{(a,a)} \vee y_{(a,c)} \vee y_{(c,a)} \vee y_{(c,c)} \\ Z(t_i, t_j) &\mapsto z_{(a,a)} \vee z_{(a,c)} \vee z_{(c,a)} \vee z_{(c,c)} \end{aligned}$$

where $x_{(a,a)}, x_{(a,c)}, x_{(c,a)}, x_{(c,c)}, y_{(a,a)}, y_{(a,c)}, y_{(c,a)}, y_{(c,c)}, z_{(a,a)}, z_{(a,c)}, z_{(c,a)}, z_{(c,c)} \in \{\ominus, \oplus, \blacklozenge, \blacktriangleleft, \blacktriangleright\}$.

As represented in Table 4.2, there are altogether eight possible termination event combinations for three transactions t_i, t_j, and t_k. These termination event combinations are set into formula (4.1).

t_i	t_k	t_j	$X(t_i,t_k) \wedge Y(t_k,t_j) \Rightarrow Z(t_i,t_j)$
a	a	a	$x_{(a,a)} \wedge y_{(a,a)} \Rightarrow z_{(a,a)}$
a	a	c	$x_{(a,a)} \wedge y_{(a,c)} \Rightarrow z_{(a,c)}$
a	c	a	$x_{(a,c)} \wedge y_{(c,a)} \Rightarrow z_{(a,a)}$
a	c	c	$x_{(a,c)} \wedge y_{(c,c)} \Rightarrow z_{(a,c)}$
c	a	a	$x_{(c,a)} \wedge y_{(a,a)} \Rightarrow z_{(c,a)}$
c	a	c	$x_{(c,a)} \wedge y_{(a,c)} \Rightarrow z_{(c,c)}$
c	c	a	$x_{(c,c)} \wedge y_{(c,a)} \Rightarrow z_{(c,a)}$
c	c	c	$x_{(c,c)} \wedge y_{(c,c)} \Rightarrow z_{(c,c)}$

Table 4.2: Termination Event Combinations for Transitive Dependencies

From the results represented in Table 4.2 we derive the following integrated formulas:

$$(x_{(a,a)} \wedge y_{(a,a)}) \vee (x_{(a,c)} \wedge y_{(c,a)}) \Rightarrow z_{(a,a)} \quad (4.2)$$
$$(x_{(a,a)} \wedge y_{(a,c)}) \vee (x_{(a,c)} \wedge y_{(c,c)}) \Rightarrow z_{(a,c)} \quad (4.3)$$
$$(x_{(c,a)} \wedge y_{(a,a)}) \vee (x_{(c,c)} \wedge y_{(c,a)}) \Rightarrow z_{(c,a)} \quad (4.4)$$
$$(x_{(c,a)} \wedge y_{(a,c)}) \vee (x_{(c,c)} \wedge y_{(c,c)}) \Rightarrow z_{(c,c)} \quad (4.5)$$

The formulas above are the basis for deriving transitive dependencies. The conjunction ($\wedge$) and disjunction ($\vee$) of two dependencies x and y is defined in Table 4.3.

x	y	$(x \wedge y)$	$(x \vee y)$	x	y	$(x \wedge y)$	$(x \vee y)$	x	y	$(x \wedge y)$	$(x \vee y)$
$\ominus$	$\oplus$	$\ominus$	$\oplus$	$\blacktriangleleft$	$\oplus$	$\blacktriangleleft$	$\oplus$	$\blacklozenge$	$\oplus$	$\blacktriangleleft$	$\oplus$
$\ominus$	$\blacktriangleright$	$\ominus$	$\blacktriangleright$	$\blacktriangleleft$	$\blacktriangleright$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacktriangleright$	$\blacklozenge$	$\blacklozenge$
$\ominus$	$\blacktriangleleft$	$\ominus$	$\blacktriangleleft$	$\blacktriangleleft$	$\blacktriangleleft$	$\blacktriangleleft$	$\blacktriangleleft$	$\blacklozenge$	$\blacktriangleleft$	$\blacktriangleleft$	$\blacklozenge$
$\ominus$	$\blacklozenge$	$\ominus$	$\blacklozenge$	$\blacktriangleleft$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$
$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\blacktriangleleft$	$\ominus$	$\ominus$	$\blacktriangleleft$	$\blacklozenge$	$\ominus$	$\ominus$	$\blacklozenge$
$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\blacktriangleright$	$\oplus$	$\ominus$	$\oplus$				
$\oplus$	$\blacktriangleright$	$\blacktriangleright$	$\oplus$	$\blacktriangleright$	$\blacktriangleright$	$\blacktriangleright$	$\blacktriangleright$				
$\oplus$	$\blacktriangleleft$	$\ominus$	$\oplus$	$\blacktriangleright$	$\blacktriangleleft$	$\ominus$	$\blacklozenge$				
$\oplus$	$\blacklozenge$	$\blacktriangleright$	$\oplus$	$\blacktriangleright$	$\blacklozenge$	$\blacktriangleright$	$\blacklozenge$				
$\oplus$	$\ominus$	$\ominus$	$\oplus$	$\blacktriangleright$	$\ominus$	$\ominus$	$\blacktriangleright$				

Table 4.3: Conjunction and Disjunction of x and y

From Table 4.3 follows that $\ominus$ is stronger than ($\blacklozenge, \blacktriangleleft, \blacktriangleright$), whereas ($\blacklozenge, \blacktriangleleft, \blacktriangleright$) are stronger than $\oplus$. However, in the case of combinations with ($\blacklozenge, \blacktriangleleft, \blacktriangleright$) we have to consider the direction of this dependency. The $\blacklozenge$-symbol can be seen as a combination of $\blacktriangleleft$ and $\blacktriangleright$. For the $\wedge$-connectivity we have to find a way along the two triangles to get a triangle as result, e.g. $\blacktriangleright$ and $\blacktriangleright$ leads to $\blacktriangleright$, whereas $\blacktriangleright$ and $\blacktriangleleft$ leads to $\ominus$. In case one of the elements is $\oplus$, the way along the plus and the triangle has to end at a triangle of the same direction, e.g. $\oplus$ and $\blacktriangleright$ leads to $\blacktriangleright$, whereas $\oplus$ and $\blacktriangleleft$ leads to $\ominus$.

The reason for these results is that the connection of two symbols by an $\wedge$-operator is a combination of a termination events of transactions t_i, t_k, and t_j. The first symbol represents the connection between the termination events of t_i and t_k and the second of t_k and t_j. Obviously, the termination event of t_k has to be the same in both cases. The combination of a $\blacktriangleright$ and $\blacktriangleleft$ (t_k is compensatable) cannot lead to one of the compensating symbols ($\blacklozenge, \blacktriangleleft, \blacktriangleright$), because neither t_i nor t_j are compensatable in this case. The resulting symbol is $\ominus$ because we have to set a stronger constraint on the termination event combination than the possibility of compensation represented by a triangle.

Using the rules represented in Table 4.3 and the formulas (4.2)–(4.5), we are able to derive the transitive dependency Z for a given dependency pair X and Y. We demonstrate this for the following example. Suppose, there are three transactions t_i, t_k, and t_j which are connected by the following dependencies:

$$vital\_dep_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep_{\blacktriangleright}(t_k, t_j)$$

Table 4.1 illustrates the symbols for the termination event combinations of the termination dependencies. Setting these symbols in the formulas (4.2)–(4.5) results in a pattern which indicates that the derived dependency is $vital\_dep_{\blacklozenge}(t_i, t_j)$:

$$\begin{aligned}
(\oplus \wedge \oplus) \vee (\ominus \wedge \ominus) &\Rightarrow \oplus \\
(\oplus \wedge \blacktriangleright) \vee (\ominus \wedge \oplus) &\Rightarrow \blacktriangleright \\
(\blacktriangleleft \wedge \oplus) \vee (\oplus \wedge \ominus) &\Rightarrow \blacktriangleleft \\
(\blacktriangleleft \wedge \blacktriangleright) \vee (\oplus \wedge \oplus) &\Rightarrow \oplus
\end{aligned}$$

In conclusion, using the rules represented in Table 4.3 and the formulas (4.2)–(4.5), we can derive for each pair of termination dependencies $X(t_i, t_k)$ and $Y(t_k, t_j)$ the corresponding transitive termination dependency $Z(t_i, t_j)$.

4.2 Computing Valid Dependency Combinations for a Transitive Dependency

The considerations in the previous section are the basis for an algorithm which determines all valid combinations of dependencies X and Y for a given transitive dependency Z. In detail, we start with the given dependency $Z(t_i, t_j)$, e.g. $vital\_dep_{\blacklozenge}(t_i, t_j)$, which represents a collection of symbols for different termination event combination, e.g. $z_{(a,a)} = \oplus$, $z_{(a,c)} = \blacktriangleright$, $z_{(c,a)} = \blacktriangleleft$, and $z_{(c,c)} = \oplus$. The following general form of the formulas (4.2)–(4.5) build the basis for the evaluation of the values of x_a, y_b, x_c, and y_d of the dependencies X and Y with a given symbol z_m:

$$((x_a \wedge y_b) \vee (x_c \wedge y_d)) \Rightarrow z_m$$

We try to find the minimal set of values (symbols) for x_a, y_b, x_c, and y_d which fulfill the formula above. Computing the transitive event combination z_m is done for each symbol separately in a step of the following algorithm:

```
procedure Deriving_Dependency_Combinations
    // Input:   Transitive Dependency Z
    // Output:  Set of Dependency Combinations X and Y which
    //          imply Z
  initialize;
  repeat
    repeat
      compute ⊖ (invalid terms);
      compute ⊕ (valid terms);
    until (no new results evaluated);
    compute ⧫ (symmetrical compensating terms);
    compute ◀ (left-hand compensating terms);
    compute ▶ (right-hand compensating terms);
  until (no new results evaluated);
  compute consistent_dependency_combinations;
end;
```

The algorithm mainly consists of a nested loop. In the inner loop the evaluation rules for valid ($z_m = \oplus$) and invalid terms ($z_m = \ominus$) are applied until no new results for x_a, y_b, x_c, and y_d can be generated. Then the rules of the compensating terms ($z_m = \blacklozenge, \blacktriangleleft$ or $\blacktriangleright$) are executed. If this leads to further results, the loop for the valid and invalid terms is executed again. After the nested loop the results have to be adjusted with respect to compensation. In the following the single steps are described in detail:

Initializing: We start with the initialization of $x_{(a,a)}$ and $y_{(a,a)}$ which are valid ($\oplus$) by definition. There is no dependency where the abortion of both transactions is not valid.

After the initialization we continue with the steps of computing single terms for the four z_m. In these steps of the algorithm we have to consider the following *general observation rules*:

1. Due to the fact that x_a represents $x_{(a,a)}$ and $x_{(c,a)}$ and the definition of the termination dependencies (cf. Table 4.1), there are only the three possible values $\ominus, \oplus, \blacktriangleleft$ for x_a. In contrast, y_b stands for $y_{(a,a)}$ and $y_{(a,c)}$. Thus, there are the following three possible values $\ominus, \oplus, \blacktriangleright$ for y_b.

2. The value $\blacklozenge$ appears only within the $exc_{\blacklozenge}$ dependency in the case that both transactions commits. Therefore, only $x_{(c,c)}$ and $y_{(c,c)}$ are candidates for the value $\blacklozenge$.

Compute Invalid Terms ($z_m = \ominus$): To guarantee that z_m is invalid, the following term

$$\underbrace{(x_a \wedge y_b) \vee (x_c \wedge y_d)}_{\ominus}$$

has to evaluated to the invalid value $\ominus$ according to the rules presented in Table 4.4[1] which is a reduced version of Table 4.3.

x	$\wedge$	y
$\ominus$	$\wedge$	$\star$
$\star$	$\wedge$	$\ominus$
$\oplus$	$\wedge$	$\blacktriangleleft$
$\blacktriangleright$	$\wedge$	$\blacktriangleleft$
$\blacktriangleright$	$\wedge$	$\oplus$
$\ominus$		

x	$\vee$	y
$\ominus$	$\vee$	$\ominus$
$\ominus$		

Table 4.4: Evaluation of Invalid Terms

From the rules in Table 4.4 follows, both terms $(x_a \wedge y_b)$ and $(x_c \wedge y_d)$ have to evaluate to a invalid value in order to get $\ominus$ as a result of the disjunction. Each of these terms is again $\ominus$, if one of its elements, e.g. x_a or y_b, is $\ominus$ or x_a is $\oplus$ and y_b is $\blacktriangleleft$. We define the following evaluation rules:

- If an element of a conjunction is $\blacklozenge$, then the other one has to be $\ominus$.
- If the first element of a term (x_a or x_c) is $\blacktriangleleft$, then the other one has to be $\ominus$.
- If the first element of a term is $\blacktriangleright$, then the other one has to be $\ominus, \oplus$ or$\blacktriangleleft$.
- If the first element of a term is $\oplus$, then the other one has to be $\ominus$ or $\blacktriangleleft$.
- If the second element of a term (y_b or y_d) is $\blacktriangleleft$, then the other one has to be $\ominus$, $\oplus$ or $\blacktriangleright$.
- If the second element of a term is $\blacktriangleright$, then the other one has to be $\ominus$.
- If the second element of a term is $\oplus$, then the other one has to be $\ominus$ or $\blacktriangleright$.
- From the general observation rules $x_a \in \{\ominus, \oplus, \blacktriangleleft\}$ and $y_b \in \{\ominus, \oplus, \blacktriangleright\}$ follows that either x_a or y_b has to be $\ominus$.

Compute Valid Terms ($z_m = \oplus$): To guarantee that z_m is valid, the term

$$\underbrace{(x_a \wedge y_b) \vee (x_c \wedge y_d)}_{\oplus}$$

x	$\wedge$	y
$\oplus$	$\wedge$	$\oplus$
	$\oplus$	

x	$\vee$	y
$\oplus$	$\vee$	$\star$
$\star$	$\vee$	$\oplus$
	$\oplus$	

Table 4.5: Evaluation of Valid Terms

has to be evaluated to the valid value $\oplus$ according to the rules presented in Table 4.5.

As represented in Table 4.5, either term $(x_a \wedge y_b)$ or term $(x_c \wedge y_d)$ have to evaluate to a valid value (one of them or both) to get $\oplus$ as a result of the disjunction. On the other hand, each term is valid, if both of its elements (x_a and y_b and x_c and y_d, respectively) are $\oplus$. Therefore, we define the following evaluation rules:

- If an element of a term is not $\oplus$, then the other term has to be $\oplus$ and, thus, both elements of this term has to be $\oplus$.
- If an element of each term is $\oplus$, then one of the other has to be $\oplus$, too.

Compute Symmetrical Compensating Terms ($z_m = \blacklozenge$): To guarantee that z_m is compensatable in both directions, the term

$$\underbrace{(x_a \wedge y_b) \vee (x_c \wedge y_d)}_{\blacklozenge}$$

has to be evaluated to the compensating value $\blacklozenge$ using the rules presented in Table 4.6.

x	$\wedge$	y
$\blacklozenge$, $\blacktriangleleft$	$\wedge$	$\blacklozenge$, $\blacktriangleright$
	$\blacklozenge$	

x	$\vee$	y
$\blacktriangleright$	$\vee$	$\blacklozenge$, $\blacktriangleleft$
$\blacktriangleleft$	$\vee$	$\blacklozenge$, $\blacktriangleright$
$\blacklozenge$	$\vee$	$\ominus$, $\blacklozenge$, $\blacktriangleleft$, $\blacktriangleright$
	$\blacklozenge$	

Table 4.6: Evaluation of a Symmetrical Compensating Term

From the results in Table 4.6 follows that one of the terms $(x_a \wedge y_b)$ and $(x_c \wedge y_d)$ has to evaluate to $\blacklozenge$ and the other one to $\blacktriangleright$, $\blacktriangleleft$, $\blacklozenge$, or $\ominus$, or the first term evaluates to $\blacktriangleleft$ ($\blacktriangleright$) and the second one to $\blacktriangleright$ ($\blacktriangleleft$). Each term is $\blacklozenge$, if the first element is $\blacktriangleleft$ or $\blacklozenge$ and the other element is $\blacklozenge$ or $\blacktriangleright$.

- If an element of a term is $\ominus$, then the other term has to be $\blacklozenge$.
- If $(x_a \wedge y_b)$ is $\blacklozenge$, then x_a is set to $\blacktriangleleft$ and y_b to $\blacktriangleright$ due to the general observation rules that $x_a \in \{\ominus, \oplus, \blacktriangleleft\}$ and $y_b \in \{\ominus, \oplus, \blacktriangleright\}$.

[1] The $\star$-symbol stands for "don't care".

- If x_a or y_d is $\oplus$, then the term evaluates to $\blacktriangleright$ or $\ominus$ and the other term has to be $\blacklozenge$.
- If y_b is $\oplus$, then the term evaluates to $\blacktriangleleft$ or $\ominus$ and the other term has to be $\blacklozenge$ or $\blacktriangleright$.
- If x_c is $\oplus$, then the term evaluates to $\blacktriangleright$ or $\ominus$ and the other term has to be $\blacklozenge$ or $\blacktriangleleft$.
- If both y_b and x_c are valid ($\oplus$), then the first term has to evaluate to $\blacktriangleleft$ (set x_a to $\blacktriangleleft$) and the second to $\blacktriangleright$ (set y_d to $\blacktriangleright$ or $\blacklozenge$).

Compute Left-hand Compensating Terms ($z_m = \blacktriangleleft$): To guarantee that z_m is valid with a compensatable transaction t_i, the term

$$\underbrace{(x_a \wedge y_b) \vee (x_c \wedge y_d)}_{\blacktriangleleft}$$

has to be evaluated to the compensating value $\blacktriangleleft$ according to the rules in Table 4.7.

x	$\wedge$	y
$\blacklozenge$, $\blacktriangleleft$	$\wedge$	$\oplus$, $\blacktriangleleft$
	$\blacktriangleleft$	

x	$\vee$	y
$\blacktriangleleft$	$\vee$	$\blacktriangleleft$, $\ominus$
$\ominus$	$\vee$	$\blacktriangleleft$
	$\blacktriangleleft$	

Table 4.7: Evaluation of a Left-hand Compensating Term

As represented in Table 4.7, one of the terms $(x_a \wedge y_b)$ and $(x_c \wedge y_d)$ has to evaluate to $\blacktriangleleft$ and the other one has to evaluate to $\blacktriangleleft$ or $\ominus$. Each term is $\blacktriangleleft$, if its first element is $\blacktriangleleft$ or $\blacklozenge$, and the second element is $\blacktriangleleft$ or $\oplus$. Consequently, we derive the following evaluation rules:

- If an element of a term is $\ominus$ and, thus, the whole term is $\ominus$, the other term has to be $\blacktriangleleft$.
- If $(x_a \wedge y_b)$ has to evaluate to $\blacktriangleleft$, then x_a has to be set to $\blacktriangleleft$ and y_b to $\oplus$ because x_a cannot evaluate to $\blacklozenge$ and y_b cannot be $\blacktriangleleft$ due to the general observation rules.
- If element x_a is $\oplus$, then the term cannot evaluate to $\blacktriangleleft$ and has to evaluate to $\ominus$. This is only possible if y_b evaluates to $\ominus$.
- If element y_b is $\oplus$, then x_a may be either $\blacktriangleleft$ or $\blacklozenge$ (if the general observation rules are fulfilled). Additionally, x_a may be $\ominus$ when the other term $(x_c \wedge y_d)$ may evaluate to $\blacktriangleleft$.
- If x_c is $\oplus$, then the term has to evaluate to $\ominus$ and y_d has to be $\ominus$ or $\blacktriangleleft$.

Compute Right-hand Compensating Terms ($z_m = \blacktriangleright$): To guarantee that z_m is valid with a compensatable transaction t_j, the term

$$\underbrace{(x_a \wedge y_b) \vee (x_c \wedge y_d)}_{\blacktriangleright}$$

has to be evaluated to the right-hand compensatable value $\blacktriangleright$ according to the rules in Table 4.8.

x	$\wedge$	y
$\oplus$, $\blacktriangleright$	$\wedge$	$\blacklozenge$, $\blacktriangleright$
	$\blacktriangleright$	

x	$\vee$	y
$\blacktriangleright$	$\vee$	$\blacktriangleright$, $\ominus$
$\ominus$	$\vee$	$\blacktriangleright$
	$\blacktriangleright$	

Table 4.8: Evaluation of a Right-hand Compensating Term

From the rules in Table 4.8 follows that one of the terms $(x_a \wedge y_b)$ and $(x_c \wedge y_d)$ has to evaluate to $\blacktriangleright$ and the other one to $\blacktriangleright$ or $\ominus$. Each term is $\blacktriangleright$, if the first element (x_a or x_c) is $\oplus$ or $\blacktriangleright$ and the other element (y_b or y_d) is $\blacklozenge$ or $\blacktriangleright$. Thus, we define the following evaluation rules:

- If an element of a term is $\ominus$, then the other term has to be $\blacktriangleright$.
- If $(x_a \wedge y_b)$ has to evaluate to $\blacktriangleright$, then x_a has to be set to $\oplus$ and y_b to $\blacktriangleright$ because x_a cannot evaluate to a $\blacktriangleleft$ and y_b not to $\blacklozenge$ due to the general observation rules.
- If element y_b is $\oplus$, then the term cannot be evaluated to $\blacktriangleright$ and has to be evaluated to $\ominus$. This is only possible if x_a is evaluated to $\ominus$.
- If element x_a is $\oplus$, then y_b may be either $\blacktriangleright$ or $\blacklozenge$ (if the general observation rules are fulfilled). Additionally, y_b may be $\ominus$ when the other term $(x_c \wedge y_d)$ may evaluate to $\blacktriangleright$.
- If y_d is $\oplus$, then the term has to evaluate to $\ominus$ and x_c has to be $\ominus$ or $\blacktriangleright$.

Compute Consistent Dependency Combinations: After the nested loop of the algorithm the results for the termination event combination of X and Y are combined to possible dependencies $X(t_i, t_k)$ and $Y(t_k, t_j)$. In this step, these set of dependencies are adjusted with respect to the compensation property. In case one of the involved transactions is compensatable, this property must be reflected in all corresponding dependencies. For example, if the transactions t_i and t_k are compensatable and vital-dependent for each other, then the related dependency is $vital\_dep_{\blacklozenge}(t_i, t_k)$ and not $vital\_dep_{\blacktriangleleft}(t_i, t_k)$, $vital\_dep_{\blacktriangleright}(t_i, t_k)$, or $vital\_dep(t_i, t_k)$. In conclusion, we can only build dependency combinations of $X(t_i, t_k)$ and $Y(t_k, t_j)$ where the compensation property of transaction t_k is reflected in both dependencies. For example, a combination of $vital\_dep_{\blacklozenge}(t_i, t_k)$ and $vital(t_k, t_j)$ is not valid. In contrast, $vital\_dep_{\blacktriangleleft}(t_i, t_k)$ and $vital(t_k, t_j)$ is a valid dependency combination. In this case t_k is non-compensatable.

The algorithm described in this section produces a set of rules (see Appendix A) including all reasonable and consistent dependency combinations.

4.3 Exemplary Application of the Algorithm

The following example shall illustrate our algorithm for evaluating valid combinations of dependencies for a given transitive dependency.

Example 4.3.1 *Let t_i, t_k, and t_j be transactions and $vital\_dep_{\blacklozenge}(t_i, t_j)$ the given dependency Z. According to the Tables 4.1, $z_{(a,a)}$ is set to $\oplus$, $z_{(a,c)}$ to $\blacktriangleright$, $z_{(c,a)}$ to $\blacktriangleleft$, and $z_{(c,c)}$ to $\oplus$. In this case we have to evaluate two valid terms, one right-hand and one left-hand compensating term. Below we demonstrate how our algorithm produces valid dependency combinations for the given transitive dependency.*

1. ***Initializing:*** *By definition the abortion of both transactions is allowed for all dependencies, i.e., $x_{(a,a)}$ and $y_{(a,a)}$ are always initialized with $\oplus$. Thus, the following formulas has to be fulfilled:*

$$(\underbrace{x_{(a,a)}}_{\oplus} \wedge \underbrace{y_{(a,a)}}_{\oplus}) \vee (x_{(a,c)} \wedge y_{(c,a)}) \quad \Rightarrow \quad \oplus$$
$$(\underbrace{x_{(a,a)}}_{\oplus} \wedge y_{(a,c)}) \vee (x_{(a,c)} \wedge y_{(c,c)}) \quad \Rightarrow \quad \blacktriangleright$$
$$(x_{(c,a)} \wedge \underbrace{y_{(a,a)}}_{\oplus}) \vee (x_{(c,c)} \wedge y_{(c,a)}) \quad \Rightarrow \quad \blacktriangleleft$$
$$(x_{(c,a)} \wedge y_{(a,c)}) \vee (x_{(c,c)} \wedge y_{(c,c)}) \quad \Rightarrow \quad \oplus$$

2. ***Compute Valid Terms:*** *Because there are no invalid terms we start with the valid terms. The first valid term directly evaluates to $\oplus$ because the terms $(x_{(a,a)} \wedge y_{(a,a)})$ evaluates to $\oplus$ and the disjunction of a $\oplus$ and another value also evaluates to $\oplus$. Without any intermediate results for $x_{(c,a)}$, $y_{(a,c)}$, $x_{(c,c)}$, and $y_{(c,c)}$, we cannot derive whether the last formulas is valid or not. Thus, there are no new results and we continue with the computation of the compensating terms.*

3. ***Compute Left-hand Compensating Term:*** *The term $(x_{(c,a)} \wedge y_{(a,a)})$ can only be evaluated to $\blacktriangleleft$, if $x_{(c,a)}$ is set to $\blacktriangleleft$ or $\blacklozenge$. However, the second general observation rule says that only $x_{(c,c)}$ and not $x_{(c,a)}$ could evaluate to $\blacklozenge$. Furthermore, $x_{(c,a)}$ may be set to $\ominus$ when the other term $(x_{(c,c)} \wedge y_{(c,a)})$ evaluates to $\blacktriangleleft$ (see the evaluation rules for left-hand compensating terms):*

$$(\underbrace{x_{(c,a)}}_{\blacktriangleleft,\ominus} \wedge \underbrace{y_{(a,a)}}_{\oplus}) \vee \underbrace{(x_{(c,c)} \wedge y_{(c,a)})}_{\blacktriangleleft,\ominus} \quad \Rightarrow \quad \blacktriangleleft$$

4. ***Compute Right-hand Compensating Term:*** *The evaluation of the right-hand compensating term is similar to the left-hand compensating term. The intermediate results are as follows:*

$$(\underbrace{x_{(a,a)}}_{\oplus} \wedge \underbrace{y_{(a,c)}}_{\blacktriangleright,\ominus}) \vee \underbrace{(x_{(a,c)} \wedge y_{(c,c)})}_{\blacktriangleright,\ominus} \quad \Rightarrow \quad \blacktriangleright$$

   We cannot derive further values for the variables of the compensating terms. Thus, we leave this loop and return to the first loop where the valid and invalid terms are evaluated again.

5. ***Compute Valid Term:*** *The disjunction of the last formula evaluates only to $\oplus$, if $(x_{(c,a)} \wedge y_{(a,c)})$ or $(x_{(c,c)} \wedge y_{(c,c)})$ evaluates to $\oplus$. However, two steps before we derived that $x_{(c,a)}$ may only be either $\blacktriangleleft$ or $\ominus$ and $y_{(a,c)}$ either $\blacktriangleright$ or $\ominus$. Thus, the conjunction $(x_{(c,c)} \wedge y_{(c,c)})$ has to be $\oplus$ (see evaluation rules for valid terms). This is done by setting both elements to $\oplus$:*

$$(\underbrace{x_{(c,a)}}_{\blacktriangleleft,\ominus} \wedge \underbrace{y_{(a,c)}}_{\blacktriangleright,\ominus}) \vee \underbrace{(\underbrace{x_{(c,c)}}_{\oplus} \wedge \underbrace{y_{(c,c)}}_{\oplus})}_{\oplus} \quad \Rightarrow \quad \oplus$$

6. ***Compute Left-hand Compensating Term:*** *We may derive further results for the left-hand compensating term by considering the values determined in the step before. The conjunction $(x_{(c,c)} \wedge y_{(c,a)})$ cannot be evaluated to $\blacktriangleleft$ because the first element $x_{(c,c)}$ is $\oplus$. Thus, this term has to evaluate to $\ominus$ and the other term $(x_{(c,a)} \wedge y_{(a,a)})$ to $\blacktriangleleft$ in order to finally derive a $\blacktriangleleft$ for the corresponding disjunction. For that $x_{(c,a)}$ is set to $\blacktriangleleft$. Furthermore, $y_{(c,a)}$ is set to $\ominus$ or $\blacktriangleleft$ to evaluate the conjunction to $\ominus$. In summary the following values are determined:*

$$\underbrace{(\underbrace{x_{(c,a)}}_{\blacktriangleleft} \wedge \underbrace{y_{(a,a)}}_{\oplus})}_{\blacktriangleleft} \vee \underbrace{(\underbrace{x_{(c,c)}}_{\oplus} \wedge \underbrace{y_{(c,a)}}_{\blacktriangleleft,\ominus})}_{\ominus} \quad \Rightarrow \quad \blacktriangleleft$$

7. ***Compute Right-hand Compensating Term:*** *The evaluation of the right-hand compensating term is again similar to the left-hand compensating term. The results are as follows:*

$$\underbrace{(\underbrace{x_{(a,a)}}_{\oplus} \wedge \underbrace{y_{(a,c)}}_{\blacktriangleright})}_{\blacktriangleright} \vee \underbrace{(\underbrace{x_{(a,c)}}_{\blacktriangleright,\ominus} \wedge \underbrace{y_{(c,c)}}_{\oplus})}_{\ominus} \quad \Rightarrow \quad \blacktriangleright$$

   The result of the nested loop of the algorithm is represented in Figure 4.2. There are two possibilities for the dependencies X and Y. The dependency X may be either $vital\_dep_{\blacktriangleleft}(t_i, t_k)$ or $vital\_dep_{\blacklozenge}(t_i, t_k)$ and the dependency Y may be either $vital\_dep_{\blacktriangleright}(t_k, t_j)$ or $vital\_dep_{\blacklozenge}(t_k, t_j)$. Thus, there are four possible dependency combinations for Z.

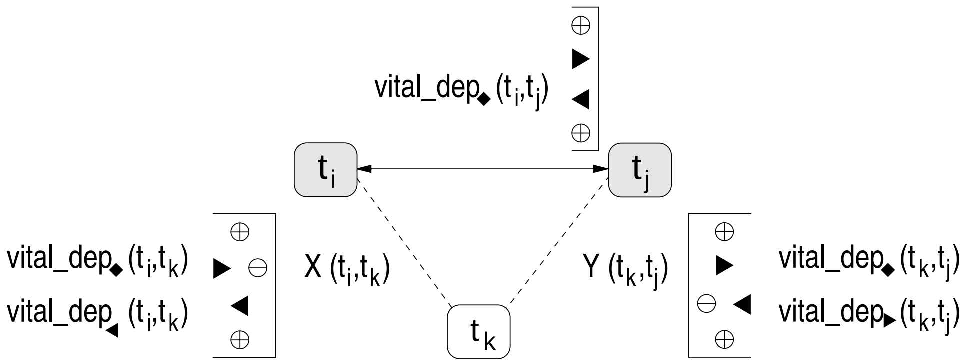

Figure 4.2: Evaluating the Transitive Dependency *vital_dep*$_\blacklozenge$

8. ***Compute Consistent Dependency Combinations:*** *The combinations vital_dep*$_\blacklozenge(t_i, t_k)$ *with vital_dep*$_\blacktriangleright(t_k, t_j)$ *and vital_dep*$_\blacktriangleleft(t_i, t_k)$ *with vital_dep*$_\blacklozenge(t_k, t_j)$ *are not valid because transaction* t_k *is contradictory specified with respect to the compensation property. In one dependency* t_k *is defined as compensatable whereas in the other dependency it is defined as non-compensatable. In conclusion, the following two valid dependency combinations are evaluated for the transitive dependency vital_dep*$_\blacklozenge(t_i, t_j)$*:*

$$\begin{aligned} vital\_dep_\blacklozenge(t_i, t_k) \quad &\wedge \quad vital\_dep_\blacklozenge(t_k, t_j) \quad \Rightarrow \quad vital\_dep_\blacklozenge(t_i, t_j) \\ vital\_dep_\blacktriangleleft(t_i, t_k) \quad &\wedge \quad vital\_dep_\blacktriangleright(t_k, t_j) \quad \Rightarrow \quad vital\_dep_\blacklozenge(t_i, t_j) \end{aligned}$$

Chapter 5

Application of the Transitivity Rules in an Example Scenario

The following examples are intended to clarify the application of transaction closures and the derivation of transitive termination dependencies in such transaction closures. The transaction closure in our first example can be considered as a workflow with special dependencies among the related transactions. Especially, there is a transaction which is executed outside the scope of its initiating transaction.

Example 5.0.2 *A commonly used example is a travel planning activity. In our example this activity consists of the recording of the customer's data, reserving a flight, booking a room in the beach hotel including a diving course and a car, or booking a room in the city hotel. This activity is modeled as transaction closure with the transactions* t_2, t_3, t_4, t_5, t_6, t_7, *and the coordinating root transaction* t_1.

Transaction t_2 *represents the recording of the customer's data which is done independently whether the trip reservation is successful or not. The flight reservation* (t_3) *is essential for the trip. Moreover, a room in the beach hotel* (t_4) *may be reserved including a diving course* (t_6). *Furthermore, we may rent a car* (t_7) *for traveling along the coast. If the diving course cannot be booked we also cannot book the beach hotel. An alternative to the beach hotel is the city hotel* (t_5).

From this scenario follows that the transactions t_2, t_3, *and* t_4 *which are child transactions of the root transaction* t_1 *are connected to* t_1 *by the following dependencies:*

$$indep(t_1, t_2) \;\wedge\; vital\_dep_{\blacktriangleleft}(t_3, t_1) \;\wedge\; vital_{\blacktriangleright}(t_1, t_4)$$

The transactions t_5, t_6, *and* t_7 *are child transactions of transaction* t_4. *These child transactions are connected to their parent transaction by the following dependencies:*

$$exc_{\blacklozenge}(t_4, t_5) \;\wedge\; vital\_dep_{\blacktriangleleft}(t_4, t_6) \;\wedge\; dep_{\blacktriangleleft}(t_4, t_7)$$

As reflected by the dependencies, transactions t_3, t_4, t_5, t_6, *and* t_7 *are supposed to be compensatable.*

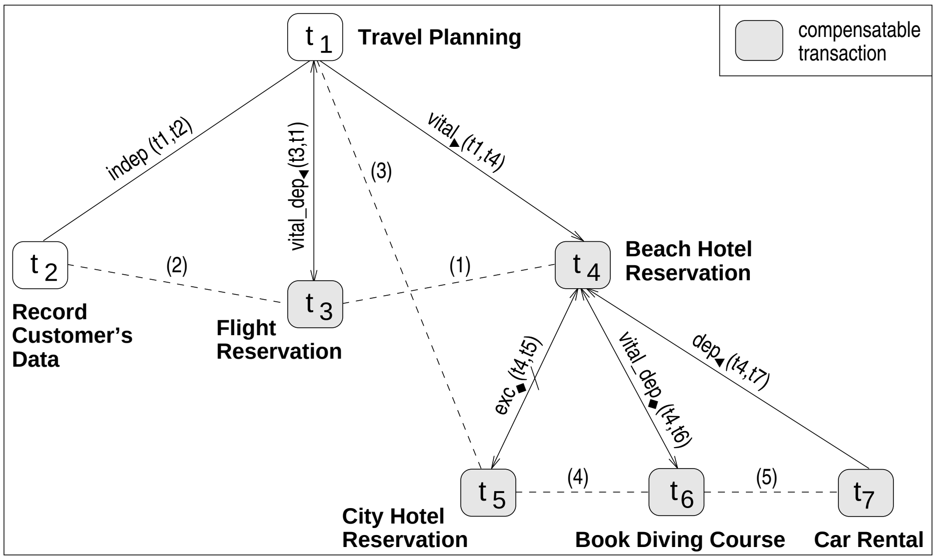

Figure 5.1: A sample Transaction Closure

Our example transaction closure is graphically illustrated in Figure 5.1 where the arrows denote the direction of the abort dependencies. For example, $t_1 \longrightarrow t_4$ *means that the abortion of transaction* t_1 *leads to the abortion of* t_4*. The termination dependencies are graphically represented as follows:*

$$\begin{aligned}
vital(t_i,t_j) &\quad \text{corresponds to} \quad t_i \longrightarrow t_j \\
dep(t_i,t_j) &\quad \text{corresponds to} \quad t_i \longleftarrow t_j \\
vital_dep(t_i,t_j) &\quad \text{corresponds to} \quad t_i \longleftrightarrow t_j \\
exc(t_i,t_j) &\quad \text{corresponds to} \quad t_i \not\longleftrightarrow t_j \\
indep(t_i,t_j) &\quad \text{corresponds to} \quad t_i \;\text{——}\; t_j
\end{aligned}$$

From our dependency definitions, Theorem 3.1.6, and Theorem 3.2.9, and the rules in Appendix A we can now derive the transitive dependencies in the underlying transaction closure. We are also able to investigate the influence of the abortion of a certain transaction on the whole closure. In the following, we discuss some interesting cases which refer to the dashed lines in Figure 5.1:

(1) We start with the consideration of the transactions t_1*,* t_3*, and* t_4*. We know that* t_1 *is* $vital\_dep_{\blacktriangleleft}$ *on* t_3 *and* $vital_{\blacktriangleright}$ *for* t_4*. From these basic dependencies we can derive that transaction* t_3 *is* transitively $vital_{\blacktriangleright}$ *for transaction* t_4 *(see Rule A.39 in Appendix A):*

$$vital\_dep_{\blacktriangleleft}(t_3,t_1) \wedge vital_{\blacktriangleright}(t_1,t_4) \stackrel{Rule\ A.39}{\Longrightarrow} vital_{\blacktriangleright}(t_3,t_4)$$

Hence, the abortion of transaction t_3 *(no flight can be reserved) leads to the abortion of the parent* t_1 *(cancel of the trip) and of the sibling transaction* t_4 *(we do not need* *a hotel reservation). In contrast, the abortion of t_4 has no influence on the other transactions. In other words, in case there is no free room in the beach hotel, nevertheless, we may book the flight.*

(2) Transaction t_2 is independent *of its initiating transaction and, thus, it may be* transitively independent *of all other transactions of the transaction closure. For example, from Rule 3.5 of Theorem 3.1.6 and Rule A.75 of Appendix A we can derive that t_2 is also independent of t_3:*

$$\begin{aligned} indep(t_1, t_2) \wedge vital\_dep_{\blacktriangleleft}(t_3, t_1) &\overset{Rule\ 3.5}{\equiv} \\ indep(t_2, t_1) \wedge vital\_dep_{\blacktriangleleft}(t_3, t_1) &\overset{Rule\ A.75}{\Longrightarrow} indep(t_2, t_3) \end{aligned}$$

(3) An other interesting dependency is the exclusive dependency between the two hotel reserving transactions t_4 and t_5. The relationship between the transactions t_5 and t_1 can be derived from the vital$_{\blacktriangleright}$ dependency between t_1 and t_4.

$$vital_{\blacktriangleright}(t_1, t_4) \wedge exc_{\blacklozenge}(t_4, t_5) \overset{Rule\ A.102}{\Longrightarrow} indep(t_1, t_5) \vee vital_{\blacktriangleright}(t_1, t_5) \vee exc_{\blacktriangleright}(t_1, t_5)$$

Here, transaction t_1 can be specified by the transaction designer as either $indep(t_1, t_5)$, $vital_{\blacktriangleright}(t_1, t_5)$, or $exc_{\blacktriangleright}(t_1, t_5)$. In this application scenario we would model this dependency as vital$_{\blacktriangleright}$. Thus, the reservation of the city hotel is canceled in case the trip reservation is canceled. This may happen if we cannot book a flight.

(4) Due to the specification that transaction t_5 is exclusive for transaction t_6, the successful reservation of the city hotel leads to the abortion of the beach hotel reservation including the diving course:

$$\begin{aligned} exc_{\blacklozenge}(t_4, t_5) \wedge vital\_dep_{\blacklozenge}(t_4, t_6) &\overset{Rule\ 3.12}{\equiv} \\ exc_{\blacklozenge}(t_5, t_4) \wedge vital\_dep_{\blacklozenge}(t_4, t_6) &\overset{Rule\ A.69}{\Longrightarrow} exc_{\blacklozenge}(t_5, t_6) \end{aligned}$$

(5) Finally, we consider transaction t_7. Transaction t_4 is dependent on t_7. Thus, a room in the beach hotel is only reserved in case we can rent a car. The dependency between the car rental and the diving course booking is evaluated by the following rules:

$$\begin{aligned} dep_{\blacktriangleleft}(t_4, t_7) \wedge vital\_dep_{\blacklozenge}(t_4, t_6) &\overset{Rule\ 3.11}{\equiv} \\ vital_{\blacktriangleright}(t_7, t_4) \wedge vital\_dep_{\blacklozenge}(t_4, t_6) &\overset{Rule\ A.69}{\Longrightarrow} vital_{\blacktriangleright}(t_7, t_6) \end{aligned}$$

Transaction t_7 is vital for t_6 and, thus, the diving course can only be booked in case a car is rent.

Example 5.0.2 showed that the abortion of a transaction may lead to the abortion of parts of the closure. Often, dependencies between two arbitrary transactions of a transaction closure may be complex and sometimes not obvious. However, the influence of transaction abortion on a transaction closure can be simulated by the transitive dependencies. This may help also to detect superfluous, redundant parts of a transaction closure definition. The following Example 5.0.3 (see also Figure 5.2) is intended to show the handling of inconsistent dependency specifications.

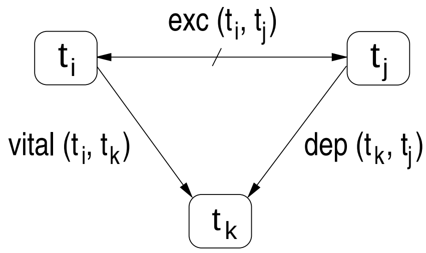

Figure 5.2: Example of a Contradicting Dependency Specification

Example 5.0.3 *Suppose there are three transactions t_i, t_j, and t_k with the following dependencies:*

$$vital(t_i, t_k) \wedge exc(t_i, t_j) \wedge dep(t_k, t_j)$$

The exclusive dependency only allows the commitment of one of the related transactions t_i and t_j or the abortion of both transactions:

$$\begin{array}{rcl} commit_{t_i} \overset{exc(t_i,t_j)}{\Rightarrow} abort_{t_j} & \overset{dep(t_j,t_k)}{\Rightarrow} & abort_{t_k} \\ commit_{t_j} \overset{exc(t_i,t_j)}{\Rightarrow} abort_{t_i} & \overset{vital(t_i,t_k)}{\Rightarrow} & abort_{t_k} \\ (abort_{t_i} \wedge abort_{t_j}) & \overset{vital(t_i,t_k)}{\Rightarrow} & abort_{t_k} \end{array}$$

In case transaction t_i commits, transaction t_j has to abort because of the exclusive dependency. However, the abortion of t_j leads to the abortion of transaction t_k. In the opposite case, if transaction t_j commits, then t_i and t_k have to abort. The case that the transactions t_i and t_j abort leads to the abortion of t_k. Due to the exclusive dependency the commitment of the transactions t_i and t_j is not allowed. As a consequence, transaction t_k is aborted in any case. Thus, there is no rule which allows to specify such a dependency combination. Rule A.97 on page 39 determines that the transitive dependency for $vital(t_i, t_k)$ and $dep(t_k, t_j)$ may be either independent, vital-dependent, vital, or dependent, but not exclusive:

$$vital(t_i, t_k) \wedge dep(t_k, t_j) \quad \Rightarrow \quad indep(t_i, t_j) \vee vital\_dep(t_i, t_j) \vee vital(t_i, t_j) \vee dep(t_i, t_j)$$

Using the set of transitivity rules we are able to detect such contradictory dependency specification.

Chapter 6

Conclusions and Outlook

The original model of nested transactions is not suitable for generalized transaction models where child transactions may survive the termination of the parent transaction. The proposed concept of transaction closures together with the dependencies introduced provide an appropriate model for discussing such extended models (as proposed in e.g. [Elm92]). Transaction closures are collections of transactions where the connection is given by termination conditions between transactions and their immediate child transactions. Compensating transactions are also part of the model. The framework allows to effectively compute derived termination conditions, and, thus, it builds the basis for transaction design tools which can help in designing less failure-prone and more efficient applications. Besides termination dependencies, our framework also supports object visibility constraints and execution order constraints [STS98].

Our future work will concentrate on the enforcement of termination conditions in generalized transaction management systems. Here, we will attempt to adopt the methods proposed in [TV95, GHK93, GD94] to our framework and provide some extensions to capture the transitive properties of transaction dependencies. Besides this, several extended transaction models like ConTracts [WR92, RSS97] or S-transaction [VEH92] will be mapped onto our framework to show the practicability of the approach. Moreover, we will use our framework for describing global transactions in federated database environments where the component systems support different transaction types. We believe that with our framework we are able to exactly formulate the different relationships of the global child transactions which are executed by the possibly heterogeneous and autonomous component database systems.

Bibliography

[BHG87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA, 1987.

[BÖH+92] A. Buchmann, M. T. Özsu, M. Hornick, D. Georgakopoulos, and F. Manola. A Transaction Model for Active Distributed Object Systems. In A. K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 123–151, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[CR90] P. K. Chrysanthis and K. Ramamritham. ACTA: A Framework for Specifying and Reasoning about Transaction Structure and Behavior. In H. Garcia-Molina and H. V. Jagadish, editors, *Proc. of the 1990 ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ*, ACM SIGMOD Record, Vol. 19, No. 2, pages 194–203, June 1990.

[CR91] P. K. Chrysanthis and K. Ramamritham. A Formalism for Extended Transaction Models. In G. M. Lohmann, A. Sernadas, and R. Camps, editors, *Proc. of the 17th Int. Conf. on Very Large Data Bases (VLDB'91), Barcelona, Spain*, pages 103–112. Morgan Kaufmann Publishers, San Mateo, CA, September 1991.

[CR94] P. K. Chrysanthis and K. Ramamritham. Synthesis of Extended Transaction Models Using ACTA. *ACM Transaction on Database Systems*, 19(3):450–491, September 1994.

[DHL91] U. Dayal, M. Hsu, and R. Ladin. A Transaction Model for Long-Running Activities. In G. M. Lohmann, A. Sernadas, and R. Camps, editors, *Proc. of the 17th Int. Conf. on Very Large Data Bases (VLDB'91), Barcelona, Spain*, pages 113–122. Morgan Kaufmann Publishers, San Mateo, CA, September 1991.

[Elm92] A. K. Elmagarmid, editor. *Database Transaction Models For Advanced Applications.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Gar83] H. Garcia-Molina. Using Semantic Knowledge for Transaction Processing in a Distributed Database. *ACM Transactions on Database Systems*, 8(2):186–213, June 1983.

[GD94] A. Geppert and K. R. Dittrich. Rule-Based Implementation of Transaction Model Specifications. In N. W. Paton and M. H. Williams, editors, *Rules in Database Systems, Proc. of the 1st Int. Workshop, RIDS'93, Edinburgh, Scotland, August 1993*, Workshops in Computing, pages 127–142. Springer-Verlag, London, 1994.

[GHK93] D. Georgakopoulos, M. Hornick, and P. Krychniak. An Environment for the Specification and Management of Extended Transactions in DOMS. In H.-J. Schek, A. P. Sheth, and B. D. Czejdo, editors, *Proc. of the 3rd Int. Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems (RIDE-IMS'93), Vienna, Austria*, pages 253–257, IEEE Computer Society Press, April 1993.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, April 1995.

[GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[HLM88] M. Hsu, R. Ladin, and D. R. McCarthy. An Execution Model For Active Data Base Management Systems. In C. Beeri, J. W. Schmidt, and U. Dayal, editors, *Proc. of the 3rd Int. Conf. on Data and Knowledge Bases: Improving Usability and Responsiveness, Jerusalem, Israel, June, 1988*, pages 171–179, Morgan Kaufmann Publishers, 1988.

[KR96] M. U. Kamath and K. Ramamritham. Correctness Issues in Workflow Management. *Distributed Systems Engineering*, 3(4), December 1996.

[Mos85] J. E. B. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing.* MIT Press, Cambridge, MA, 1985.

[RC97] K. Ramamritham and P. K. Chrysanthis. *Advances in Concurrency Control and Transaction Processing.* IEEE Computer Society Press, Los Alamitos, California, 1997.

[RKT+95] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wäsch, and P. Muth. Towards a Cooperative Transaction Model — The Cooperative Activity Model. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proc. of the 21st Int. Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland*, pages 194–205, Morgan Kaufmann Publishers, September 1995.

[RSS97] A. Reuter, K. Schneider, and F. Schwenkreis. ConTracts Revisited. In S. Jajodia and L. Kerschberg, editors, *Advanced Transaction Models and Architectures*, chapter 5, Kluwer Academic Publishers, 1997.

[STS98] K. Schwarz, C. Türker, and G. Saake. Analyzing and Formalizing Dependencies in Generalized Transaction Structures. In *Proc. of the Int. Workshop on Issues and Applications of Database Technology (IADT'98), July 6-9, 1998, Berlin, Germany*, 1998. *To appear.*

[TV95] J. Tang and J. Veijalainen. Enforcing Inter-task Dependencies in Transactional Workflows. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95), May 9–12, 1995, Vienna, Austria*, pages 72–86, 1995.

[VEH92] J. Veijalainen, F. Eliassen, and B. Holtkamp. The S-transaction Model. In A. K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 467–513, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[WR92] H. Wächter and A. Reuter. The ConTract Model. In A. K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 219–263, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

Appendix A

Rules for Transitive Transaction Termination Dependencies

The algorithm presented in Chapter 4 brings out a set of rules for deriving transitive dependencies. There are some cases where on the right side of the implication a disjunction of possible dependencies occur, i.e, from two given dependencies we cannot determine only exactly one correct transitive dependency. These are the cases where we — as transaction designer — have the option to choose among several possible dependencies. The corresponding transitivity rules are computed by considering the three dependencies X, Y and Z in all possible combinations $(X \wedge Y \Rightarrow Z)$, $(Y \wedge Z \Rightarrow X)$, and $(X \wedge Z \Rightarrow Y)$ and under the requirement that the resulting rule set must be consistent.

In the following, we list the resulting transitive dependency rules which have to hold for three different transactions t_i, t_k, and t_j:

Rules for $vital\_dep(t_i, t_j)$**:**

$$vital\_dep(t_i, t_k) \quad \wedge \quad vital\_dep(t_k, t_j) \quad \Rightarrow \quad vital\_dep(t_i, t_j) \tag{A.1}$$
$$vital\_dep_{\blacktriangleright}(t_i, t_k) \quad \wedge \quad vital\_dep_{\blacktriangleleft}(t_k, t_j) \quad \Rightarrow \quad vital\_dep(t_i, t_j) \tag{A.2}$$

Rules for $vital(t_i, t_j)$**:**

$$vital(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.3}$$
$$vital(t_i, t_k) \quad \wedge \quad vital\_dep(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.4}$$
$$vital\_dep(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.5}$$
$$vital_{\blacktriangleright}(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.6}$$
$$vital_{\blacktriangleright}(t_i, t_k) \quad \wedge \quad vital\_dep_{\blacktriangleleft}(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.7}$$
$$vital\_dep_{\blacklozenge}(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.8}$$
$$vital\_dep_{\blacktriangleleft}(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.9}$$
$$vital\_dep_{\blacktriangleright}(t_i, t_k) \quad \wedge \quad vital(t_k, t_j) \quad \Rightarrow \quad vital(t_i, t_j) \tag{A.10}$$

Rules for $dep(t_i, t_j)$**:**

$$\begin{aligned}
dep(t_i, t_k) &\wedge dep(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.11)}\\
vital_dep(t_i, t_k) &\wedge dep(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.12)}\\
dep(t_i, t_k) &\wedge vital_dep(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.13)}\\
dep(t_i, t_k) &\wedge dep_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.14)}\\
vital_dep_{\blacktriangleright}(t_i, t_k) &\wedge dep_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.15)}\\
dep(t_i, t_k) &\wedge vital_dep_{\blacklozenge}(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.16)}\\
dep(t_i, t_k) &\wedge vital_dep_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.17)}\\
dep(t_i, t_k) &\wedge vital_dep_{\blacktriangleright}(t_k, t_j) &&\Rightarrow dep(t_i, t_j) && \text{(A.18)}
\end{aligned}$$

Rules for $exc(t_i, t_j)$**:**

$$\begin{aligned}
vital_dep(t_i, t_k) &\wedge exc(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.19)}\\
exc(t_i, t_k) &\wedge vital_dep(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.20)}\\
dep(t_i, t_k) &\wedge exc(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.21)}\\
exc(t_i, t_k) &\wedge vital(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.22)}\\
vital_dep_{\blacktriangleright}(t_i, t_k) &\wedge exc_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.23)}\\
exc_{\blacktriangleright}(t_i, t_k) &\wedge vital_dep_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.24)}\\
dep(t_i, t_k) &\wedge exc_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.25)}\\
exc_{\blacktriangleright}(t_i, t_k) &\wedge vital(t_k, t_j) &&\Rightarrow exc(t_i, t_j) && \text{(A.26)}
\end{aligned}$$

Rules for $vital\_dep_{\blacktriangleleft}(t_i, t_j)$**:**

$$\begin{aligned}
vital_dep_{\blacktriangleleft}(t_i, t_k) &\wedge vital_dep(t_k, t_j) &&\Rightarrow vital_dep_{\blacktriangleleft}(t_i, t_j) && \text{(A.27)}\\
vital_dep_{\blacklozenge}(t_i, t_k) &\wedge vital_dep_{\blacktriangleleft}(t_k, t_j) &&\Rightarrow vital_dep_{\blacktriangleleft}(t_i, t_j) && \text{(A.28)}
\end{aligned}$$

Rules for $vital\_dep_{\blacktriangleright}(t_i, t_j)$**:**

$$\begin{aligned}
vital_dep(t_i, t_k) &\wedge vital_dep_{\blacktriangleright}(t_k, t_j) &&\Rightarrow vital_dep_{\blacktriangleright}(t_i, t_j) && \text{(A.29)}\\
vital_dep_{\blacktriangleright}(t_i, t_k) &\wedge vital_dep_{\blacklozenge}(t_k, t_j) &&\Rightarrow vital_dep_{\blacktriangleright}(t_i, t_j) && \text{(A.30)}
\end{aligned}$$

Rules for $vital\_dep_{\blacklozenge}(t_i, t_j)$**:**

$$\begin{aligned}
vital_dep_{\blacklozenge}(t_i, t_k) &\wedge vital_dep_{\blacklozenge}(t_k, t_j) &&\Rightarrow vital_dep_{\blacklozenge}(t_i, t_j) && \text{(A.31)}\\
vital_dep_{\blacktriangleleft}(t_i, t_k) &\wedge vital_dep_{\blacktriangleright}(t_k, t_j) &&\Rightarrow vital_dep_{\blacklozenge}(t_i, t_j) && \text{(A.32)}
\end{aligned}$$

Rules for $vital_{\blacktriangleright}(t_i, t_j)$**:**

$$\begin{aligned}
vital(t_i, t_k) &\wedge vital_{\blacktriangleright}(t_k, t_j) &&\Rightarrow vital_{\blacktriangleright}(t_i, t_j) && \text{(A.33)}\\
vital(t_i, t_k) &\wedge vital_dep_{\blacktriangleright}(t_k, t_j) &&\Rightarrow vital_{\blacktriangleright}(t_i, t_j) && \text{(A.34)}
\end{aligned}$$

$$vital\_dep(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.35}$$
$$vital_{\blacktriangleright}(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.36}$$
$$vital_{\blacktriangleright}(t_i, t_k) \wedge vital\_dep_{\blacklozenge}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.37}$$
$$vital\_dep_{\blacklozenge}(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.38}$$
$$vital\_dep_{\blacktriangleleft}(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.39}$$
$$vital\_dep_{\blacktriangleright}(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow vital_{\blacktriangleright}(t_i, t_j) \tag{A.40}$$

Rules for $dep_{\blacktriangleleft}(t_i, t_j)$:

$$dep_{\blacktriangleleft}(t_i, t_k) \wedge dep(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.41}$$
$$vital\_dep_{\blacktriangleleft}(t_i, t_k) \wedge dep(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.42}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.43}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge dep_{\blacktriangleleft}(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.44}$$
$$vital\_dep_{\blacklozenge}(t_i, t_k) \wedge dep_{\blacktriangleleft}(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.45}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep_{\blacklozenge}(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.46}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep_{\blacktriangleleft}(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.47}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep_{\blacktriangleright}(t_k, t_j) \Rightarrow dep_{\blacktriangleleft}(t_i, t_j) \tag{A.48}$$

Rules for $exc_{\blacktriangleleft}(t_i, t_j)$:

$$vital\_dep_{\blacktriangleleft}(t_i, t_k) \wedge exc(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.49}$$
$$exc_{\blacktriangleleft}(t_i, t_k) \wedge vital\_dep(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.50}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge exc(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.51}$$
$$exc_{\blacktriangleleft}(t_i, t_k) \wedge vital(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.52}$$
$$vital\_dep_{\blacklozenge}(t_i, t_k) \wedge exc_{\blacktriangleleft}(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.53}$$
$$exc_{\blacklozenge}(t_i, t_k) \wedge vital\_dep_{\blacktriangleleft}(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.54}$$
$$dep_{\blacktriangleleft}(t_i, t_k) \wedge exc_{\blacktriangleleft}(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.55}$$
$$exc_{\blacklozenge}(t_i, t_k) \wedge vital(t_k, t_j) \Rightarrow exc_{\blacktriangleleft}(t_i, t_j) \tag{A.56}$$

Rules for $exc_{\blacktriangleright}(t_i, t_j)$:

$$vital\_dep(t_i, t_k) \wedge exc_{\blacktriangleright}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.57}$$
$$exc(t_i, t_k) \wedge vital\_dep_{\blacktriangleright}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.58}$$
$$dep(t_i, t_k) \wedge exc_{\blacktriangleright}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.59}$$
$$exc(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.60}$$
$$vital\_dep_{\blacktriangleright}(t_i, t_k) \wedge exc_{\blacklozenge}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.61}$$
$$exc_{\blacktriangleright}(t_i, t_k) \wedge vital\_dep_{\blacklozenge}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.62}$$
$$dep(t_i, t_k) \wedge exc_{\blacklozenge}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.63}$$
$$exc_{\blacktriangleright}(t_i, t_k) \wedge vital_{\blacktriangleright}(t_k, t_j) \Rightarrow exc_{\blacktriangleright}(t_i, t_j) \tag{A.64}$$

Rules for $exc_{\blacklozenge}(t_i, t_j)$:

$$\mathit{vital\_dep}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{exc}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.65}$$

$$\mathit{exc}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{vital\_dep}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.66}$$

$$\mathit{dep}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{exc}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.67}$$

$$\mathit{exc}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{vital}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.68}$$

$$\mathit{vital\_dep}_{\blacklozenge}(t_i,t_k) \wedge \mathit{exc}_{\blacklozenge}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.69}$$

$$\mathit{exc}_{\blacklozenge}(t_i,t_k) \wedge \mathit{vital\_dep}_{\blacklozenge}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.70}$$

$$\mathit{dep}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{exc}_{\blacklozenge}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.71}$$

$$\mathit{exc}_{\blacklozenge}(t_i,t_k) \wedge \mathit{vital}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{exc}_{\blacklozenge}(t_i,t_j) \tag{A.72}$$

Rules for $indep(t_i, t_j)$:

$$\mathit{indep}(t_i,t_k) \wedge \mathit{vital\_dep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.73}$$

$$\mathit{indep}(t_i,t_k) \wedge \mathit{vital\_dep}_{\blacklozenge}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.74}$$

$$\mathit{indep}(t_i,t_k) \wedge \mathit{vital\_dep}_{\blacktriangleleft}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.75}$$

$$\mathit{indep}(t_i,t_k) \wedge \mathit{vital\_dep}_{\blacktriangleright}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.76}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{vital}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}(t_i,t_j) \vee \\ & \mathit{exc}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleleft}(t_i,t_j)\end{aligned} \tag{A.77}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{vital}_{\blacktriangleright}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}_{\blacktriangleright}(t_i,t_j) \vee \\ & \mathit{exc}_{\blacklozenge}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleright}(t_i,t_j)\end{aligned} \tag{A.78}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{dep}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{dep}(t_i,t_j) \vee \\ & \mathit{dep}_{\blacktriangleleft}(t_i,t_j)\end{aligned} \tag{A.79}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{dep}_{\blacktriangleleft}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{dep}(t_i,t_j) \vee \\ & \mathit{dep}_{\blacktriangleleft}(t_i,t_j)\end{aligned} \tag{A.80}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{exc}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}(t_i,t_j) \vee \\ & \mathit{exc}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleleft}(t_i,t_j)\end{aligned} \tag{A.81}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{exc}_{\blacklozenge}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}_{\blacktriangleright}(t_i,t_j) \vee \\ & \mathit{exc}_{\blacklozenge}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleright}(t_i,t_j)\end{aligned} \tag{A.82}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{exc}_{\blacktriangleleft}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}(t_i,t_j) \vee \\ & \mathit{exc}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleleft}(t_i,t_j)\end{aligned} \tag{A.83}$$

$$\begin{aligned}\mathit{indep}(t_i,t_k) \wedge \mathit{exc}_{\blacktriangleright}(t_k,t_j) \Rightarrow\ & \mathit{indep}(t_i,t_j) \vee \mathit{vital}_{\blacktriangleright}(t_i,t_j) \vee \\ & \mathit{exc}_{\blacklozenge}(t_i,t_j) \vee \mathit{exc}_{\blacktriangleright}(t_i,t_j)\end{aligned} \tag{A.84}$$

$$\mathit{vital\_dep}(t_i,t_k) \wedge \mathit{indep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.85}$$

$$\mathit{vital\_dep}_{\blacklozenge}(t_i,t_k) \wedge \mathit{indep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.86}$$

$$\mathit{vital\_dep}_{\blacktriangleleft}(t_i,t_k) \wedge \mathit{indep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.87}$$

$$\mathit{vital\_dep}_{\blacktriangleright}(t_i,t_k) \wedge \mathit{indep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \tag{A.88}$$

$$\mathit{vital}(t_i,t_k) \wedge \mathit{indep}(t_k,t_j) \Rightarrow \mathit{indep}(t_i,t_j) \vee \mathit{vital}(t_i,t_j) \vee$$

$$vital_{\blacktriangleright}(t_i, t_j) \tag{A.89}$$

$$vital_{\blacktriangleright}(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital(t_i, t_j) \vee \\ vital_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.90}$$

$$dep(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.91}$$

$$dep_{\blacktriangleleft}(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep_{\blacktriangleleft}(t_i, t_j) \vee \\ exc_{\blacklozenge}(t_i, t_j) \vee exc_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.92}$$

$$exc(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.93}$$

$$exc_{\blacklozenge}(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep_{\blacktriangleleft}(t_i, t_j) \vee \\ exc_{\blacklozenge}(t_i, t_j) \vee exc_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.94}$$

$$exc_{\blacktriangleleft}(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep_{\blacktriangleleft}(t_i, t_j) \vee \\ exc_{\blacklozenge}(t_i, t_j) \vee exc_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.95}$$

$$exc_{\blacktriangleright}(t_i, t_k) \wedge indep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.96}$$

$$vital(t_i, t_k) \wedge dep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital\_dep(t_i, t_j) \vee \\ vital\_dep_{\blacklozenge}(t_i, t_j) \vee vital\_dep_{\blacktriangleleft}(t_i, t_j) \vee \\ vital\_dep_{\blacktriangleright}(t_i, t_j) \vee vital(t_i, t_j) \vee \\ vital_{\blacktriangleright}(t_i, t_j) \vee dep(t_i, t_j) \vee \\ dep_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.97}$$

$$vital_{\blacktriangleright}(t_i, t_k) \wedge dep_{\blacktriangleleft}(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital\_dep(t_i, t_j) \vee \\ vital\_dep_{\blacklozenge}(t_i, t_j) \vee vital\_dep_{\blacktriangleleft}(t_i, t_j) \vee \\ vital\_dep_{\blacktriangleright}(t_i, t_j) \vee vital(t_i, t_j) \\ vital_{\blacktriangleright}(t_i, t_j) \vee dep(t_i, t_j) \vee \\ dep_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.98}$$

$$vital(t_i, t_k) \wedge exc(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.99}$$

$$vital(t_i, t_k) \wedge exc_{\blacktriangleright}(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital_{\blacktriangleright}(t_i, t_j) \vee \\ exc_{\blacklozenge}(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.100}$$

$$vital_{\blacktriangleright}(t_i, t_k) \wedge exc_{\blacktriangleleft}(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleleft}(t_i, t_j) \end{array} \tag{A.101}$$

$$vital_{\blacktriangleright}(t_i, t_k) \wedge exc_{\blacklozenge}(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee vital_{\blacktriangleright}(t_i, t_j) \vee \\ exc_{\blacklozenge}(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.102}$$

$$exc(t_i, t_k) \wedge dep(t_k, t_j) \Rightarrow \begin{array}{l} indep(t_i, t_j) \vee dep(t_i, t_j) \vee \\ exc(t_i, t_j) \vee exc_{\blacktriangleright}(t_i, t_j) \end{array} \tag{A.103}$$

$$exc_{\blacktriangleleft}(t_i, t_k) \wedge dep(t_k, t_j) \Rightarrow indep(t_i, t_j) \vee dep_{\blacktriangleleft}(t_i, t_j) \vee$$

$$
exc_{\blacklozenge}(t_i,t_j) \vee exc_{\blacktriangleleft}(t_i,t_j) \tag{A.104}
$$

$$
\begin{aligned}
exc_{\blacklozenge}(t_i,t_k) \quad \wedge \quad dep_{\blacktriangleleft}(t_k,t_j) \quad &\Rightarrow \quad indep(t_i,t_j) \vee dep_{\blacktriangleleft}(t_i,t_j) \vee \\
&\qquad exc_{\blacktriangleleft}(t_i,t_j) \vee exc_{\blacklozenge}(t_i,t_j)
\end{aligned} \tag{A.105}
$$

$$
\begin{aligned}
exc_{\blacktriangleright}(t_i,t_k) \quad \wedge \quad dep_{\blacktriangleleft}(t_k,t_j) \quad &\Rightarrow \quad indep(t_i,t_j) \vee dep \vee \\
&\qquad exc(t_i,t_j) \vee exc_{\blacktriangleright}(t_i,t_j)
\end{aligned} \tag{A.106}
$$

Furthermore, there are combinations of dependencies $X(t_i,t_k)$ and $Y(t_k,t_j)$ which do not imply any restrictions for the transitive dependency $Z(t_i,t_j)$. In this case any of the dependencies introduced so far can be valid for $Z(t_i,t_j)$. This is expressed by the predicate $any(t_i,t_j)$:

$$dep(t_i,t_k) \quad \wedge \quad vital(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.107}$$

$$exc(t_i,t_k) \quad \wedge \quad exc(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.108}$$

$$dep(t_i,t_k) \quad \wedge \quad vital_{\blacktriangleright}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.109}$$

$$dep_{\blacktriangleleft}(t_i,t_k) \quad \wedge \quad vital(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.110}$$

$$dep_{\blacktriangleleft}(t_i,t_k) \quad \wedge \quad vital_{\blacktriangleright}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.111}$$

$$exc(t_i,t_k) \quad \wedge \quad exc_{\blacktriangleright}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.112}$$

$$exc_{\blacklozenge}(t_i,t_k) \quad \wedge \quad exc_{\blacklozenge}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.113}$$

$$exc_{\blacklozenge}(t_i,t_k) \quad \wedge \quad exc_{\blacktriangleleft}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.114}$$

$$exc_{\blacktriangleleft}(t_i,t_k) \quad \wedge \quad exc(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.115}$$

$$exc_{\blacktriangleleft}(t_i,t_k) \quad \wedge \quad exc_{\blacktriangleright}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.116}$$

$$exc_{\blacktriangleright}(t_i,t_k) \quad \wedge \quad exc_{\blacklozenge}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.117}$$

$$exc_{\blacktriangleright}(t_i,t_k) \quad \wedge \quad exc_{\blacktriangleleft}(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.118}$$

$$indep(t_i,t_k) \quad \wedge \quad indep(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.119}$$

$$any(t_i,t_k) \quad \wedge \quad Y(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.120}$$

$$X(t_i,t_k) \quad \wedge \quad any(t_k,t_j) \quad \Rightarrow \quad any(t_i,t_j) \tag{A.121}$$