Issues on EMV2 propagation paths
Consistency between definitions and legality, consistency and semantics rules

Denis Buzdalov, Sergey Zelenov
ISPRAS

December 30, 2016

1 Introduction

When considering particular properties and correctness of error propagation paths in EMV2, several questions occur.

Not all terms are clearly defined and some of notions that can be derived from the text, induce doubts on whether what was meant.

2 General terms

Considering a single error propagation path, characteristics of the *source* (outgoing) propagation point will be denoted by the index "$_s$" and *destination*'s incoming propagation point by "$_d$".

Propagation points can be of *error propagation* and *error containment* types. The error type set of the outgoing (source) error propagation will be denoted by p_s, and the error type set of the outgoing error containment will be denoted by c_s. Similarly, the error type set of the incoming (destination) error propagation will be denoted by p_d, and the error type set of the incoming error containment will be denoted by c_d.

Annex says about *unspecified* error types that can or cannot be propagated by a component. They will be denoted by u_s and u_d for the source and destination side respectively. The meaning of these notions will be discussed later. Until discussed, u_s and u_d will be used as some set of error types in expressions.

3 Formalization

Having denotions from above, we will formalize legality, consistency and semantic rules from the section "Error Propagation Paths and User-defined Propagation Points and Paths". Described sets are depicted on the figure 1.

Since some of statements express obligations and some express possibilities (deontic modality), we will use standard modal operators $\Box$ and $\Diamond$ to express it. So, $\Box\varphi$ means "φ **must** hold" and $\Diamond\varphi$ means "φ **may** hold", i.e. "it is **possible** for φ to hold". It is important to understand that for any φ the following equivalency is hold: $\Diamond\varphi \equiv \neg\Box\neg\varphi$, where $\neg$ is a logical negation.

Rule	Formalization
Error propagation and error containment definition consequence	$\Box p_s \cap c_s = \varnothing$ $\Box p_d \cap c_d = \varnothing$
Legality rules	
The error type set of the outgoing error propagation must be contained in the error type set of the incoming error propagation.	$\Box p_s \subseteq p_d$
The error type set of the incoming error containment declaration must be contained in the error type set of the outgoing error containment declaration.	$\Box c_s \supseteq c_d$
The direction of the error propagation or error containment for the source must be outgoing and for the destination must be incoming.	cannot be expressed using p_s, p_d, c_s and etc.
Consistency rules	
The error type set for the error propagation source of an error propagation path must not intersect with the error type set of the destination error containment declaration or with *or* the set of unspecified error propagation types.	$\Box p_s \cap c_d = \varnothing$ $\Box p_s \cap u_d = \varnothing$
The set of unspecified error propagation types of an error propagation path source must not intersect with the error type set of the destination error containment declaration or the set of unspecified error propagation types.	$\Box u_s \cap c_d = \varnothing$ $\Box u_s \cap u_d = \varnothing$
The destination of an error propagation path is robust against unintended error propagations if the type set of its incoming error propagation declaration contains the error type set of the source error propagation, error containment declaration, and any unspecified error propagation type.	$\Diamond p_d \supseteq p_s$ $\Diamond p_d \supseteq c_s$ $\Diamond p_d \supseteq u_s$
Semantic rules	
The first rule shows that it is acceptable when a source indicates it does not intend to propagate an error of a certain type and the destination indicates it does not expect such as error type or the destination is *silent* regarding a known error type, i.e., it has not specified an error propagation or error containment for the given type.	$\Box c_s \subseteq c_d \cup u_d$
The second rule indicates that it is acceptable for the destination to indicate that it expects error of a given type, while the source indicates that it does not intend to propagate errors of the same type.	$\Diamond c_s \cap p_d \neq \varnothing$
The third rule indicates that it is acceptable for the destination to indicate that it expects error of a given type, and the source indicates error propagation of the same type or nothing is specified for the given error type.	$\Box p_s \cup u_s \supseteq p_d$
The fourth rule indicates that it is not acceptable for the destination to indicate that it does not expect errors of a given type, while the source indicates that it intends to propagate such errors or is *silent* with respect to that error type.	$\Box (p_s \cup u_s) \cap c_d = \varnothing$
The fifth rule indicates that it is not acceptable for the destination to be *silent* on the propagation of a known error type and the source to indicate propagation or also be *silent*.	$\Box u_d \cap (p_s \cup u_s) = \varnothing$

Notice, that we have emphasized the word *silent* above. It is not said what exactly is means, but we understand it as if *"unspecified"* notion was used there.

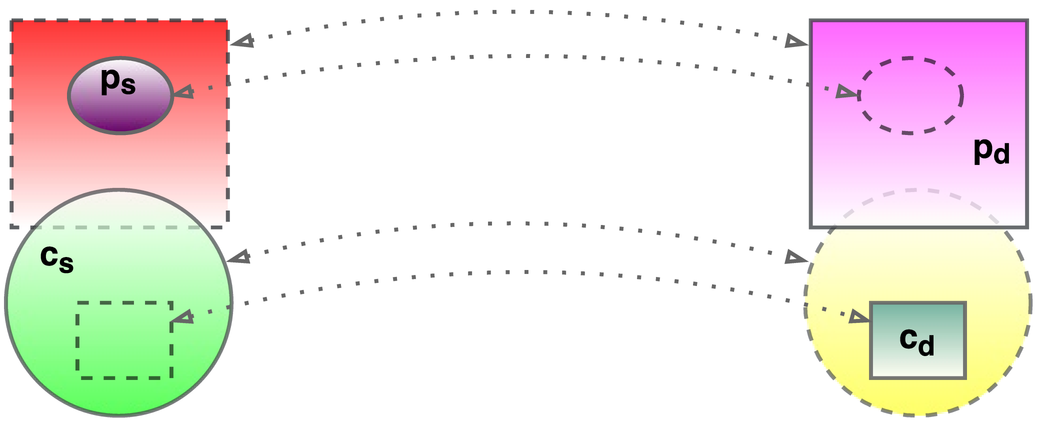

Figure 1: General view of discussed sets

4 "Unspecified" interpretations

4.1 The problem

We have realized that what is meant under the "unspecified error propagation types set" is not clear if we talk about it more or less formally. Moreover, depending on formalization, there are some contradictions between rules.

After that, we brought up five variants of possible formal definitions for u_s and u_d denoting unspecified error propagation types set at source and destination sides. These variants are pretty different: some of them are intuitive, some of them are rather simple, some of them do eliminate all contradictions with rules, some of them are symmetrical.

We are going to present all of them in the order of their creation with description of their properties. After all, we will put them in a summarizing table.

4.2 Designations

We will denote by $\mathbb{U}$ a (somewhat phantom) *universum set* of error types. Its meaning is a potential whole set of possible error types.

To eliminate misunderstandings, an operation of set difference between sets α and β will be denoted by $\alpha \setminus \beta$.

4.3 Intuitive definitions

Initially we were thinking of "unspecified" as of "those other, that are not specified", i.e. those error types that are not present in the error propagations or error containment types. It can be formalized using the universum set: $u_s = \mathbb{U} \setminus (p_s \cup c_s)$ and $u_d = \mathbb{U} \setminus (p_d \cup c_d)$. This definition is complex because of usage of $\mathbb{U}$, but it is not the only problem. First of all, such definition contradicts with the second consistency rule. Also, it really disturbs to satisfy the third semantics rule, because some finite set must include some possibly infinite or unrelated set.

To fix this complexity, we thought what if "unspecified sets" are all finite and defined in the following way: $u_s = p_d \setminus p_s$ and $u_d = c_s \setminus c_d$. On the figure 1 u_s is *red* (without a *dark-purple* part) **including** the intersection with the *green* set, and u_d is *yellow* (without a *dark-green* part) **including** the intersection with the *purple* set. This looks exactly as the previous one except excluding error types unrelated to the error propagation path (i.e., "unspecified" has path-local meaning). This approach gives much simplified definition, unfortunately still contradictory.

Such definition contradicts with the second consistency, the second semantic and the fifth semantic rules.

So, unfortunately, none of definitions that we could imagine using general understanding of the “unspecified” word were consistent with defined rules.

4.4 Rules-consistent definitions

Then, we were trying to make adequate definitions, satisfying all the rules. At the first time, we could not get rid of $\mathbb{U}$ and thus, we have got $u_s = p_d \setminus p_s$ and $u_d = \mathbb{U} \setminus (p_d \cup c_d)$ (i.e., we have changed only u_d). This definition is consistent, but it is not symmetric — “unspecified” sets on different ends of an error propagation path are defined differently.

After that, we were thinking how to get rid of $\mathbb{U}$, and we have got the following: $u_s = p_d \setminus p_s$ and $u_d = c_s \setminus (p_d \cup c_d)$. Again, we have changed only u_d with the similar principle of getting rid of $\mathbb{U}$: definition of unspecified error types was made to be path-local. On the figure 1 u_d is *yellow* (without a *dark-green* part) **excluding** the intersection with the *purple* set. This definition became simpler and all sets are finite but it is not symmetric either.

By the way, it is easy to see that using these definitions, all consistency and semantic rules can be derived from legality rules and from definitions of error propagation and error containment. Thus, it means that consistency and semantic rules actually do not add any additional limitations or explanations.

4.5 Symmetrization

The last thing we have done is we have tried to make the last simple and consistent definition to be symmetric. So, we have got $u_s = p_d \setminus (p_s \cup c_s)$ and $u_d = c_s \setminus (p_d \cup c_d)$. At this time we have changed only u_s according to u_d definition. Such way of definition is simple and symmetrical. The second semantic rule is the only thing the definition contradicts with.

4.6 Summary

To sum up all the variants described above, they were put with their characteristics in the table below.

Formalization	Finite	Symmetrical	Consistent
$u_s = \mathbb{U} \setminus (p_s \cup c_s)$ $u_d = \mathbb{U} \setminus (p_d \cup c_d)$	no	yes	no
$u_s = p_d \setminus p_s$ $u_d = c_s \setminus c_d$	yes	yes	no
$u_s = p_d \setminus p_s$ $u_d = \mathbb{U} \setminus (p_d \cup c_d)$	no	no	yes
$u_s = p_d \setminus p_s$ $u_d = c_s \setminus (p_d \cup c_d)$	yes	no	yes
$u_s = p_d \setminus (p_s \cup c_s)$ $u_d = c_s \setminus (p_d \cup c_d)$	yes	yes	no (to be discussed)

So, depending on whether symmetry is important or not, we would suggest the fourth or the fifth definition. If symmetry is not so important, the fourth definition gives the best variant since it is simple and consistent. If symmetry is important, we would propose to remove the second semantic rule and to use the fifth definition to make the whole system consistent.

Of course, we believe there are some other definitions and interpretations. If there are some, we would suggest to start the discussion to eliminate contradictions in the standard using some ways other, than we have proposed.

5 Conclusion

There are some possible contradictions or misinterpretations in the EMV2 standard due to lack of formalization. We tried to formalize some part of the standard to be able to find out what is wrong. It was found that some notions that are widely used are specified with bad precision.

We tried to eliminate possible contradictions proposing several formalizations.

We are calling for a discussion on this topic and to make a decision on particular contradictions. Additionally, we are pointing on the general problem of misformalizations in the standard and propose to work more on it.