УДК 004.056.5, 004.94

ПРИЁМЫ ОПИСАНИЯ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ ОССН ASTRA LINUX SPECIAL EDITION НА ФОРМАЛИЗОВАННОМ ЯЗЫКЕ МЕТОДА Event-B ДЛЯ ОБЕСПЕЧЕНИЯ ЕЁ ВЕРИФИКАЦИИ ИНСТРУМЕНТАМИ Rodin И ProB

П. Н. Девянин, М. А. Леонова

ООО «РусБИТех-Астра», г. Москва, Россия

Рассматриваются приёмы по доработке описания модели управления доступом отечественной защищённой операционной системы специального назначения Astra Linux Special Edition (МРОСЛ ДП-модели) в формализованной нотации (на формализованном языке метода Event-B), основанные на использовании нескольких глобальных типов, разделении общих тотальных функций на частные тотальные функции и сокращении числа инвариантов и охранных условий событий, предполагающих перебор подмножеств некоторого множества. Результатом их применения стало упрощение автоматизированной дедуктивной верификации модели с применением инструмента Rodin и её адаптация к верификации с использованием инструмента проверки моделей ProB. Данные приёмы могут быть полезны при разработке других моделей управления доступом и их верификации с применением соответствующих инструментов.

Ключевые слова: *модель управления доступом, дедуктивная верификация, Event-B, Rodin, метод проверки моделей, ProB.*

DOI 10.17223/20710410/52/5

THE TECHNIQUES OF FORMALIZATION OF OS ASTRA LINUX SPECIAL EDITION ACCESS CONTROL MODEL USING Event-B FORMAL METHOD FOR VERIFICATION USING Rodin AND ProB

P. N. Devyanin, M. A. Leonova

RusBITech-Astra, Moscow, Russia

E-mail: pdevyanin@astralinux.ru, mleonova@astralinux.ru

The paper presents techniques to specification access control model of OS Astra Linux Special Edition (the MROSL DP-model) in the formalized notation (formalized using the Event-B formal method), that are based on the use of several global types, separation of general total functions into specific total functions, reduction in the number of invariants and guard of events, which iterate over subsets of a certain set. The result of using these techniques was the simplification of automated deductive verification of formalized notation using the Rodin tool and adaptation of the model to verification by model checking formalized notation using the ProB tool. These techniques can be useful in development of the MROSL DP-model, and also in development of other access control models and verification using appropriate tools.

Keywords: *access control model, deductive verification, Event-B, Rodin, model checking, ProB.*

1. Анализ приёмов, использующихся для иерархического представления МРОСЛ ДП-модели и её верификации

Верификация модели управления доступом отечественной защищённой операционной системы специального назначения (ОССН) Astra Linux Special Edition [1, 2] необходима как для применения в основе разработки ОССН научно обоснованных технологий, так и для обеспечения выполнения при сертификации ОССН требований высшего первого уровня доверия согласно утверждённым Приказом ФСТЭК России № 76 от 02.06.2020 «Требованиям по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий» [3].

Превысивший пятьсот страниц объём описания модели, называемой мандатной сущностно-ролевой ДП-моделью безопасности управления доступом и информационными потоками в ОС семейства Linux (сокращённо МРОСЛ ДП-моделью) [4, 5], на языке, принятом в математике (в математической нотации), потребовал поиска путей повышения эффективности разработки, проверки корректности и отсутствия ошибок в самой модели. Найденным здесь решением стал перевод описания модели на формализованный язык метода Event-B (в формализованную нотацию) [6] и её автоматизированная дедуктивная верификация с применением инструмента Rodin [7]. При этом элементам, задающим в рамках МРОСЛ ДП-модели состояния системы, в формализованной нотации ставятся в соответствие переменные Event-B, правилам перехода из состояний в состояния — события Event-B, а инварианты на переменные описывают свойства внутренней согласованности элементов МРОСЛ ДП-модели [8]. В результате при помощи инструмента Rodin доказывается, что любой переход из состояния в состояние, заданный событиями Event-B, сохраняет все инварианты состояния, что позволяет убедиться в корректности описания модели и доказать выполнение в её рамках условий безопасности системы.

В то же время, несмотря на достигнутые с момента начала в 2012 г. разработки МРОСЛ ДП-модели результаты по её формированию и внедрению в ОССН, она продолжает развиваться и дорабатываться для, с одной стороны, учёта в модели изменений, вносимых в механизмы защиты ОССН, а с другой — для создания на перспективу условий для включения в ОССН новых механизмов защиты или расширения охватываемых ими компонент ОССН. Например, в модель были добавлены запрещающие роли (т. е. роли, наличие прав доступа у которых не разрешает, а, наоборот, запрещает предоставление субъект-сессиям соответствующих доступов) [9]. Также в неё были включены элементы, позволяющие моделировать управление доступом штатной СУБД PostgreSQL [10].

Таким образом, сложность и постоянное развитие модели обусловило постановку задачи по исследованию и поиску приёмов, направленных, во-первых, на согласованное описания МРОСЛ ДП-модели в математической и формализованной нотациях с целью учёта постоянно вносимых в модель изменений; во-вторых, на сокращение затрачиваемых на её верификацию ресурсов, а именно: упрощение дедуктивной верификации модели с применением поддерживающего язык метода Event-B инструмента Rodin и автоматизированное выявление при этом возможных ошибок или их дублирования; в-третьих, на обеспечение более точного соответствия модели её реализации непосредственно в программном коде ОССН.

Основой для поиска путей согласованного описания МРОСЛ ДП-модели в математической и формализованной нотациях, а также «приближения» этого описания к реализации модели в программном коде ОССН стал переход из её монолитного в иерар-

хическое представление [4, 11]. Для ОССН в иерархическом представлении МРОСЛ ДП-модели имеются четыре уровня (для СУБД PostgreSQL сформированы отдельные аналогичные четыре уровня):

— первый (базовый) — модель системы ролевого (дискреционного) управления доступом;
— второй — модель системы ролевого управления доступом и мандатного контроля целостности;
— третий — модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом только с информационными потоками по памяти;
— четвёртый — модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом с информационными потоками по памяти и по времени.

Аналогично при описании модели на формализованном языке метода Event-B используется техника пошагового уточнения (*refinement*) [12], когда вместо создания монолитной спецификации уточнение позволяет разрабатывать серию связанных между собой спецификаций, где каждая последующая спецификация в серии является уточнением предыдущих. Таким образом, в формализованной нотации модель также представляется четырьмя соответствующими описанию в математической нотации уровнями уточнений [8]. При этом сам переход на применение пошагового уточнения стал существенным шагом в развитии технологий и практических приёмов разработки МРОСЛ ДП-модели и её дедуктивной верификации.

Однако накопление опыта описания МРОСЛ ДП-модели в математической и формализованной нотациях, её дедуктивной верификации показало наличие существенных недостатков у использовавшихся для этого подходов [13]. Во-первых, это независимое друг от друга описание элементов модели для различных видов управления доступом, во-вторых, многократное дублирование одинаковых условий их выполнения, не соответствующее реализации проверок этих условий непосредственно в программном коде ОССН, в-третьих, сложность добавления уровней уточнений, моделирующих взаимодействующие с ОССН системы (например, СУБД PostgreSQL).

Для устранения этих недостатков, которые наиболее существенны для формализованной нотации, в [13] предложено логически объединить проверки условий, заданных для мандатных управления доступом, контроля целостности и ролевого управления доступом, улучшить структуру применённых при этом элементов нотации. Кроме того, для повышения качества описания и верификации МРОСЛ ДП-модели в формализованной нотации, расширения спектра применяемых для этого методов и инструментов, моделирования и в перспективе автоматизированного тестирования на соответствие этой модели её реализации непосредственно в программном коде и настройках конфигурации механизма управления доступом ОССН в настоящее время осуществляется верификация модели с использованием инструмента проверки моделей ProB [14].

Как показал опыт, используемые ProB (как и другими инструментами проверки моделей) алгоритмы перебора значений элементов верифицируемой модели и, как следствие, наличие проблемы комбинаторного взрыва в большинстве случаев также требуют доработки описания модели в формализованной нотации. То есть некоторые способы представления МРОСЛ ДП-модели на формализованной языке метода Event-B, успешно использовавшиеся при её дедуктивной верификации инструментом Rodin, оказались непригодными для применения инструмента ProB, так как приводи-

ли к завершению его работы с ошибкой вида `timeout`, вызванной превышением установленного для выполнения переборных алгоритмов интервала времени. При этом в ходе экспериментов по многократному увеличению значения данного интервала времени результат оставался одним и тем же, из чего был сделан вывод, что проблема не в задаваемом интервале, а в том, что инструмент ProB сталкивается по сути с неразрешимой для него задачей.

К приводящим к ошибкам вида `timeout` способам можно отнести использование для большинства элементов модели (субъект-сессий, сущностей, ролей и др.) единого глобального типа. К таким типам в формализованном языке метода Event-B относятся задаваемые в контексте (*context*) несущие множества (*carrier sets*) [6]. При описании элементов модели в контексте или машине (*machine*) с использованием конкретного глобального типа задаются его подмножества, например для областей значений и определения функций. При этом инструментами Rodin и, что особенно важно, ProB каждый элемент модели представляется на основе именно его глобального типа, а не используемых при описании этого элемента подмножеств глобального типа, которые для упрощения дедуктивной верификации авторами было предложено использовать в качестве подтипов [13] (подтипами будем называть подмножества глобального типа, для каждой пары которых они либо не пересекаются, либо один подтип включает другой подтип пары; подтипы позволяют выполнять над ними операции объединения, пересечения, разности, дополнения). Поэтому инструментом ProB перебор значений элементов модели предположительно выполняется на множестве всех возможных значений глобального типа этого элемента. Таким образом, при проверке инструментом ProB выполнения инвариантов (*invariants*) или охранных условий (*guards* или *grd*) событий (*events*) модели на таких множествах значений глобальных типов выполняется перебор элементов областей определения или значения функций, а если при этом использованы подмножества соответствующих областей, то и их перебор.

В результате для устранения ошибки вида `timeout` и верификации МРОСЛ ДП-модели инструментом проверки моделей ProB представление модели на формализованном языке метода Event-B было доработано авторами. Для этого разработаны и апробированы два приёма, основанные на использовании, во-первых, нескольких глобальных типов и частных тотальных функций, во-вторых, на сокращении числа инвариантов и охранных условий, предполагающих перебор подмножеств некоторого множества. Рассмотрим и проанализируем эти приёмы подробнее.

2. Использование нескольких глобальных типов и частных тотальных функций

В МРОСЛ ДП-модели для первого уровня (ролевого управления доступом) [4] в формализованной нотации, описанной в [13], задано четыре глобальных типа:

- $Names$ — множество допустимых имён сущностей, ролей, запрещающих и административных ролей;
- $Accesses$ — множество видов доступа;
- $AccessRights$ — множество видов прав доступа;
- $Union$ — множество, включающее все остальные элементы модели (субъект-сессии, сущности, роли и т. д.).

Для уменьшения мощности множества, на котором ведётся перебор значений элементов, авторами предлагается приём по разделению там, где это возможно, общего глобального типа на несколько глобальных типов. Поскольку над глобальными типами правилами языка метода Event-B не допускаются операции объединения, пересечения,

разности, дополнения (какие-либо дополнительные ограничения на эти множества могут быть заданы аксиомами — *axioms* или *axm*), то в первую очередь требуется проанализировать на предмет возможности внесения изменений, необходимых при разделении общего глобального типа, все компоненты контекстов и машин модели. К их числу относятся: несущие множества, константы (*constants*), аксиомы, переменные (*variables* или *var*), инварианты, события, включая их параметры (*parameters*), охранные условия и действия (*action* или *act*).

В результате анализа принято решение о разделении общего глобального типа *Union*, в который входили подтипы сущностей, ролей, субъект-сессий, учётных записей пользователей, на несколько глобальных типов:

— *EntitiesU* — для сущностей;
— *RolesU* — для ролей, запрещающих и административных ролей;
— *SubjectsU* — для субъект-сессий;
— *UsersU* — для учётных записей пользователей.

При этом задание нескольких глобальных типов не исключает использования их подтипов (см. листинг 1). Это связано с тем, что при применении подтипов есть ряд описанных в [13] преимуществ, в том числе: более ясное моделирование соответствующих элементов модели в формализованной нотации, расширение возможности Rodin по проверке корректности их использования, лучшая структурированность и приспособленность модели к добавлению новых уровней уточнений или модификации существующих.

```
sets
UsersU, SubjectsU, EntitiesU, RolesU, Names, Accesses,
    AccessRights
axioms
UsersUIsFinite: finite(UsersU)
SubjectsUIsFinite: finite(SubjectsU)
EntitiesUIsFinite: finite(EntitiesU)
RolesUIsFinite: finite(RolesU)
NamesIsFinite: finite(Names)
EntitiesUPartition: partition(EntitiesU, ObjectsU,
    ContainersU)
RolesUPartition: partition(RolesU, AdmRolesU, OrdRolesU,
    NRolesU)
```

Листинг 1. Задание глобальных типов с использованием подтипов

Кроме того, в формализованной нотации, представленной в [8], некоторые функции реализовывались несколькими охранными условиями (*grd*) в большинстве событий, включая события получения доступов к сущностям или ролям, изменения прав доступа ролей и ряде других. Такое представление функций обладает рядом недостатков, проанализированных в [13], в том числе: «громоздкость» и трудночитаемость охранных условий, из-за чего в них легко не заметить ошибки; повторение этих «громоздких» условий в нескольких событиях, что может дублировать уже допущенную ошибку или внести новую при редактировании условия для конкретного события; возможное несоответствие охранного условия по сути тому, как аналогичные проверки реализуются в механизме управления доступом ОССН. В связи с этим предложен подход по представлению функций математической нотации в виде тотальных функций (*total function*) в формализованной нотации, состоящих из двух видов инвариантов:

— *инвариант-типа*, задающий области определения и значения функции;
— *инвариант-истинности*, задающий условия истинности функции для элементов её области определения.

Для каждой функции инвариант-типа должен быть один, так как области определения и значения задаются однозначно, а инвариантов-истинности может быть несколько, так как для разных элементов области определения возможны различные условия истинности значения тотальной функции.

Например, в формализованной нотации модели используется $CheckRight$ — тотальная функция наличия прав доступа, параметрами которой являются субъект-сессия, сущность (роль или субъект-сессия) и право доступа, а значением — множество текущих у субъект-сессии ролей, имеющих заданное право доступа к сущности (роли или субъект-сессии).

При использовании в формализованной нотации модели общего глобального типа $Union$ инвариант-типа $CheckRightType$ тотальной функции $CheckRight$, согласно [13], имел вид, представленный в листинге 2.

```
   CheckRightType:
CheckRight ∈ (Subjects ↔ (Entities ∪ Roles ∪ Subjects ↔ AccessRights)) →
ℙ(Roles)
   CheckRightFuncE:
∀ s,e,ar,r · s ∈ Subjects ∧ e ∈ Entities ∧ ar ∈ AccessRights ∧ r ∈ Roles ⇒
(r ∈ CheckRight({s ↦ {e ↦ ar}}) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
e ↦ ar ∈ RoleRights(r))
   CheckRightFuncR:
∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Roles ∧ ar ∈ AccessRights ∧ r ∈ Roles ⇒
(r ∈ CheckRight({s ↦ {e ↦ ar}}) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
r ∈ AdmRoles ∧ e ↦ ar ∈ RoleAdmRights(r))
   CheckRightFuncS1:
∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Subjects ∧ ar ∈ AccessRights ∧
r ∈ OrdRoles ∪ AdmRoles ⇒
(r ∈ CheckRight({s ↦ {e ↦ ar}}) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
ar = Own ∧ r = SubjectOwner(e))
   CheckRightFuncS2:
∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Subjects ∧ ar ∈ AccessRights ∧ r ∈ NRoles ⇒
(r ∈ CheckRight({s ↦ {e ↦ ar}}) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
ar = Own ∧ r ∈ SubjectNOwners(e))
```

Листинг 2. Инвариант-типа и инварианты-истинности тотальной функции $CheckRight$

Однако при разделении общего глобального типа $Union$ на глобальные типы $EntitiesU$, $RolesU$, $SubjectsU$ и $UsersU$, а следовательно, невозможности объединения их подмножеств, стало необходимым разделение области определения функции $CheckRight$. В итоге задаются следующие три частные тотальные функции (листинг 3):

— $CheckRightE$ — функция наличия прав доступа у субъект-сессии к сущности;
— $CheckRightR$ — функция наличия прав доступа у субъект-сессии к роли;
— $CheckRightS$ — функция наличия прав доступа у субъект-сессии к субъект-сессии.

```
   CheckRightEType:
CheckRightE ∈ Subjects × Entities × AccessRights → ℙ(Roles)
```

```
   CheckRightEFunc:
∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Entities ∧ ar ∈ AccessRights ∧ r ∈ Roles ⇒
(r ∈ CheckRightE(s ↦ e ↦ ar) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
e ↦ ar ∈ RoleRights(r))
   CheckRightRType:
CheckRightR ∈ Subjects × Roles × AccessRights → ℙ(AdmRoles)
   CheckRightRFunc:
∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Roles ∧ ar ∈ AccessRights ∧ r ∈ AdmRoles ⇒
(r ∈ CheckRightR(s ↦ e ↦ ar) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
e ↦ ar ∈ RoleAdmRights(r))
   CheckRightSType:
CheckRightS ∈ Subjects × Subjects × {Own} → ℙ(Roles)
   CheckRightSFunc1:
∀s,e,r · s ∈ Subjects ∧ e ∈ Subjects ∧ r ∈ OrdRoles ∪ AdmRoles ⇒
(r ∈ CheckRightS(s ↦ e ↦ Own) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
r ∈ SubjectOwner(e))
   CheckRightSFunc2:
∀s,e,r · s ∈ Subjects ∧ e ∈ Subjects ∧ r ∈ NRoles ⇒
(r ∈ CheckRightS(s ↦ e ↦ Own) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
r ∈ SubjectNOwners(e))
```

Листинг 3. Задание частных тотальных функций *CheckRightE*, *CheckRightR* и *CheckRightS*

Приём по разделению общей тотальной функции *CheckRight* на частные необходим не только для корректного их задания при использовании нескольких глобальных типов. Разделение (там, где это возможно) на непересекающиеся множества областей определения и значения функций и задание на них отдельных функций позволяет упростить их переопределения в событиях, где производится перебор значений элементов этих областей.

Например, при переопределении функции *CheckRight* [13] в событии создания субъект-сессией объекта *create_object* (листинг 4) отдельным охранным условием (*grd*25) было необходимо для всех субъект-сессий s, сущностей (кроме создаваемого объекта *object*), ролей, субъект-сессий e и видов прав доступа ar значение новой функции *checkRight* задать его повторением от исходной функции *CheckRight*. При этом при работе инструмента *ProB* необходимо перебрать значения элементов всей области определения ($Entities \cup Roles \cup Subjects$), тогда как изменения вносятся только в множество сущностей *Entities* добавлением в неё нового объекта. При использование трёх функций *CheckRightE*, *CheckRightR* и *CheckRightS* необходимо переопределение только функции *CheckRightE* (листинг 5), а следовательно, сокращается перебор значений элементов.

```
   grd24:
checkRight ∈ (Subjects ↔ ((Entities ∪ {object}) ∪ Roles ∪ Subjects
↔ AccessRights)) → ℙ(Roles)
   grd25:
∀s,e,ar·s∈Subjects ∧ e ∈ Entities ∪ Roles ∪ Subjects ∧ ar ∈ AccessRights
⇒ checkRight({s ↦ {e ↦ ar}}) = CheckRight({s ↦ {e ↦ ar}})
```

```
   grd26:
∀s,ar,r · s ∈ Subjects ∧ ar ∈ AccessRights ∧ r ∈ Roles ∧
r ↦ ReadA ∉ SubjectAdmAccesses(s) ⇒ r ∉ checkRight({s ↦ {object ↦ ar}})
   grd27:
∀s,ar,r · s ∈ Subjects ∧ ar ∈ AccessRights ∧ r ∈ Roles ∧
r ↦ ReadA ∈ SubjectAdmAccesses(s) ⇒
(r ∈ checkRight({s ↦ {object ↦ ar}}) ⇔ object ↦ ar ∈ roleRights(r))
   act8:
CheckRight := checkRight
```

Листинг 4. Переопределение тотальной функции *CheckRight* в событии *create_object*

```
   grd20:
checkRightE ∈ Subjects × (Entities ∪ {object}) × AccessRights → ℙ(Roles)
   grd21:
∀s,e,ar · s ∈ Subjects ∧ e ∈ Entities ∧ ar ∈ AccessRights ⇒
checkRightE(s ↦ e ↦ ar) = CheckRightE(s ↦ e ↦ ar)
   grd22:
∀s,ar,r · s ∈ Subjects ∧ ar ∈ AccessRights ∧ r ∈ Roles ⇒
(r ∈ checkRightE(s ↦ object ↦ ar) ⇔ r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧
object ↦ ar ∈ roleRights(r))
   act7:
CheckRightE := checkRightE
```

Листинг 5. Переопределение частной тотальной функции *CheckRightE* в событии *create_object*

Дополнительным преимуществом использования частных тотальных функций взамен общих является возможность более детального их задания, что также сокращает перебор значений элементов. Сравним, например, инварианты-истинности функций *CheckRight* и *CheckRightS* (см. листинги 2 и 3). Согласно МРОСЛ ДП-модели, субъект-сессия с помощью текущей роли может обладать к другой субъект-сессии только правом доступа владения own_r (*Own*), но, используя общую тотальную функцию *CheckRight*, необходимо также задавать значения функции (пустое множество) и для остальных прав доступа, соответственно ProB при этом будет совершать дополнительный перебор. С применением частных тотальных функций на этапе задания инварианта-типа для *CheckRightS* соответствующее модели ограничение на права доступа субъект-сессии к субъект-сессии накладывается более точно.

3. Сокращение числа инвариантов и охранных условий, предполагающих перебор подмножеств некоторого множества

Ещё один приём, предлагаемый для устранения ошибки вида `timeout` и успешного применения инструмента проверки моделей ProB при верификации МРОСЛ ДП-модели, — сокращение числа инвариантов и охранных условий, предполагающих при использовании ProB перебор подмножеств некоторого множества, путём введения (там, где это возможно) тотальных функций, значениями которых являются данные подмножества. Для таких функций большинство их значений не должно изменяться в событиях, а значит, не потребуется соответствующий перебор подмножеств (например, вместо перебора подмножеств ролей для построения множества ролей, подчинённых

некоторой роли в иерархии, может быть задана функция потомков ролей), иначе использование данных функций может усложнить работу ProB.

Для примера рассмотрим изменение инварианта-истинности булевой функции доступа субъект-сессии к сущностям в контейнерах $ExecuteContainer$, параметрами которой являются субъект-сессия и сущность, а значение по определению является истинным в случае, когда в иерархии сущностей существует путь к заданной в параметрах сущности от некоторой корневой сущности-контейнера и субъект-сессия через её текущие роли имеет право доступа на выполнение $execute_r$ ($Execute$) ко всем сущностям-контейнерам, из которых состоит данный путь, и, наоборот, не имеет запрещающей роли, обладающей правом доступа $execute_r$ ($Execute$) хотя бы к одной сущности-контейнеру этого пути. В формализованной нотации, описанной в [13], данная функция представлена в виде тотальной функции $ExecuteContainer$ (листинг 6).

```
   ExecuteContainerType:
ExecuteContainer ∈ (Subjects ↔ Entities) → BOOL
   ExecuteContainerFunc:
∀s,e · s ∈ Subjects ∧ e ∈ Entities ⇒ (ExecuteContainer({s ↦ e}) = TRUE ⇔
(∃E, c · E ⊆ Containers ∧ Root ∉ E ∧ ((e ∈ dom(EntityNames) ∧
c∈dom(EntityNames(e)) ∧ Parent[E]∪{c} = E∪{Root}) ∨ (E = ∅ ∧ e = Root))
∧ (∀o · o ∈ E ∪ {Root} ⇒ (∃r · r ∈ CheckRight({s ↦ {o ↦ Execute}}) ∧
CheckRight({s ↦ {o ↦ Execute}}) ⊆ OrdRoles ∪ AdmRoles))))
```

Листинг 6. Задание тотальной функции $ExecuteContainer$

В инварианте-истинности функции ($ExecuteContainerFunc$) требуется существование подмножества множества сущностей-контейнеров E, представляющего собой путь к сущности e от некоторой корневой сущности-контейнера $Root$. При верификации модели с использованием ProB в событиях, где происходит переопределение функции $ExecuteContainer$, инструменту необходимо для каждой сущности e найти данное подмножество E путём перебора подмножеств множества глобального типа $Union$.

Данного перебора можно избежать, задав отдельно функцию $CPath$, значением которой для сущности-контейнера c является путь до неё от корневой сущности-контейнера, включая саму сущность-контейнер c (листинг 7).

```
   CPathType:
CPath ∈ Containers → ℙ1(Containers)
   CPath1:
∀c · c ∈ Containers ⇒ (c = Root ∧ CPath(c) = {Root}) ∨
(c ≠ Root ∧ {c,Root} ⊆ CPath(c) ∧ CPath(c) = CPath(Parent(c)) ∪ {c})
   NoCyclesForContainers:
∀c1,c2 · c1 ∈ Containers ∧ c2 ∈ Containers ∧ c2 ∈ CPath(c1) ∧ c1 ≠ c2 ⇒
c1 ∉ CPath(c2)
```

Листинг 7. Задание тотальной функции $CPath$

Эта функция переопределяется только в событиях создания сущности-контейнера $create\_container$ и удаления сущности $delete\_entity$, при этом исключается перебор подмножеств множества глобального типа $EntitiesU$ (функция $CPath$ была задана после разделения общего глобального типа $Union$), так как изменения вносятся однозначно для конкретной сущности-контейнера:

— при создании сущности-контейнера (каталога) *container* в сущности-контейнере (каталоге) *parent*: $CPath(container) = CPath(parent) \cup \{container\}$;
— при удалении сущности-контейнера (каталога) *container* в событии происходит проверка того, что она не содержит в себе других сущностей (каталог является пустым), а значит, *container* принадлежит только множеству $CPath(container)$ и просто исключается из области определения функции.

При использовании функции $CPath$ инвариант-истинности функции $ExecuteContainer$ имеет вид, представленный в листинге 8.

```
   ExecuteContainerFunc:
∀s,e · s ∈ Subjects ∧ e ∈ Entities ⇒ (ExecuteContainer(s ↦ e) = TRUE ⇔
e = Root ∨ (e ≠ Root ∧ (∃c · c ∈ Containers ∧ c ∈ dom(EntityNames(e)) ∧
(∀o·o∈CPath(c) ⇒ CheckRightE(s↦o↦Execute)∈ℙ1(OrdRoles∪AdmRoles)))))
```

Листинг 8. Инвариант-истинности тотальной функций *ExecuteContainer* с использованием тотальной функции $CPath$

При переопределении функции *ExecuteContainer* и нахождении её значения для субъект-сессии s и сущности e вместо перебора подмножеств множества глобального типа $Union$ для нахождения пути до сущности e используется значение функции $CPath(p)$, где p является сущностью-контейнером — родителем сущности e.

4. Апробация предложенных приёмов

Описанные приёмы прошли апробацию при верификации инструментом ProB первого уровня (уровня ролевого управления доступом) МРОСЛ ДП-модели в формализованной нотации. Без использования этих приёмов ошибка вида `timeout` возникала в следующих двух случаях.

Первый случай — при попытке инструментом ProB нахождения множества удовлетворяющих охранным условиям значений параметров для событий, в которых несколько охранных условий (*grd*) предполагали перебор подмножеств некоторого множества (например, в событии создания субъект-сессии *create_subject*) (рис. 1).

Второй случай — при задании уточнённого для инструмента ProB контекста модели (выполнении специального события *SETUP_CONTEXT*, осуществляющего инициализацию начального состояния системы в рамках модели) (рис. 2). Связано это с тем, что для верификации модели с использованием ProB (в отличие от Rodin, для которого необходимо для каждого глобального типа задать только условие его конечности) в контексте требуется явно указать мощность каждого множества глобального типа, что на практике часто является сложной задачей. С одной стороны, мощность каждого такого множества желательно задать достаточной для приближения модели в формализованной нотации к реальной ОССН, а именно: включения в него всех важных с точки зрения безопасности компонент системы. Например, множество сущностей глобального типа *EntitiesU* должно включать корневой каталог файловой системы («/»), каталог, где находятся параметры ОССН («/etc/»), домашний каталог пользователя («/home/») и др. С другой стороны, приходится учитывать накладываемые самим инструментом ProB ограничения по его производительности ввиду того, что при увеличении мощности множеств глобальных типов увеличивается и выполняемый инструментом перебор значений элементов данных множеств, что может привести к комбинаторному взрыву и завершению работы ProB с ошибкой вида `timeout`.

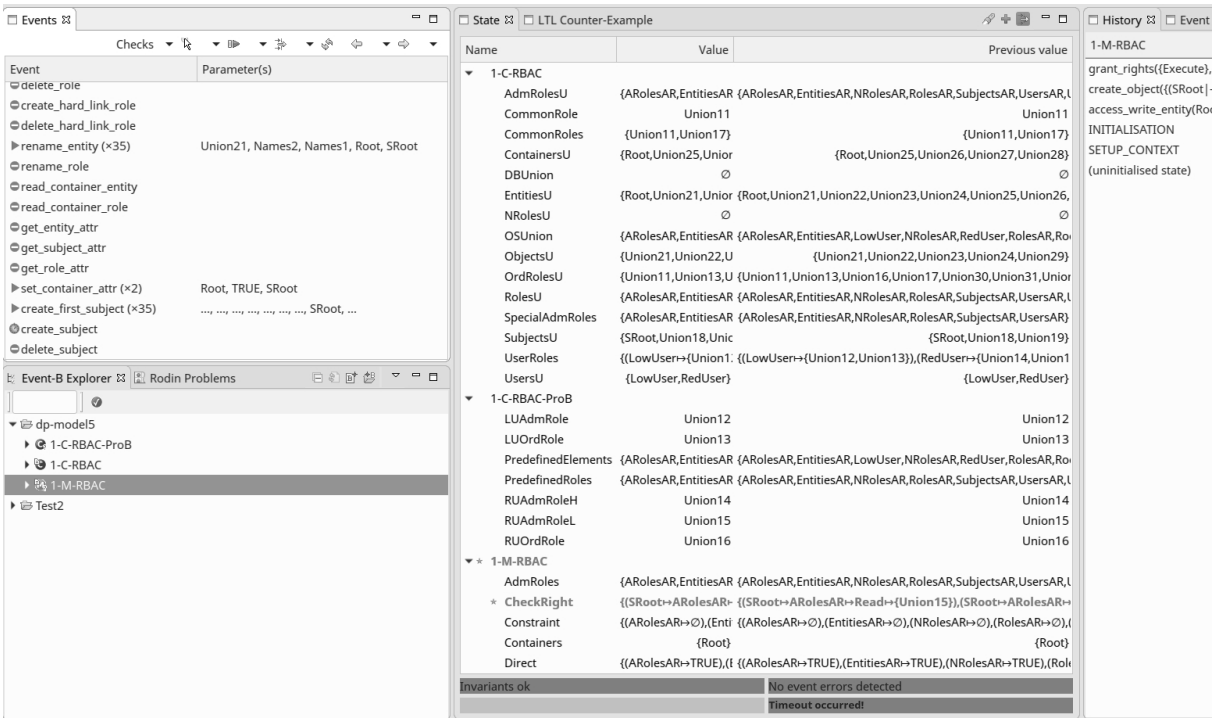

Рис. 1. Ошибка вида `timeout` для события *create_subject*

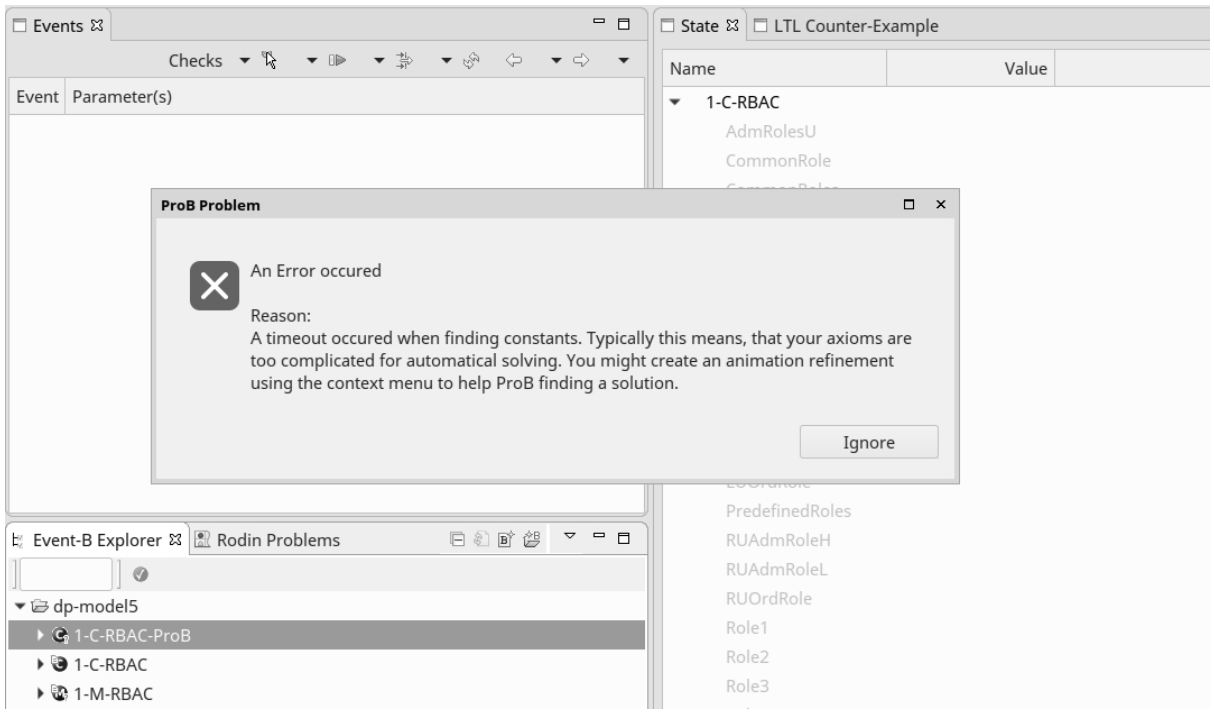

Рис. 2. Ошибка вида `timeout` при выполнении специального события *SETUP_CONTEXT*

В результате использования описанных приёмов в рассмотренных двух случаях была устранена ошибка вида `timeout`. Во-первых, стало возможным выполнить любое событие модели, так как для него алгоритмами инструмента ProB за приемлемое время происходит перебор и нахождение множества значений, удовлетворяющих охранным условиям события (пример результатов выполнения события *create_subject* приведён на рис. 3). Во-вторых, стало успешным выполнение специального события

SETUP_CONTEXT при большем числе элементов верифицируемой модели. Если ранее на развёрнутом авторами стенде данное событие выполнялось без ошибки максимум при 20 элементах в каждом состоянии системы (для $card(Union) = 20$), что очевидно недостаточно для адекватного моделирования основных важных с точки зрения безопасности компонент ОССН, то с использованием предложенных приёмов общее число элементов увеличилось до 68 (для $card(UsersU) = 10$, $card(EntitiesU) = 18$, $card(RolesU) = 30$ и $card(SubjectsU) = 10$).

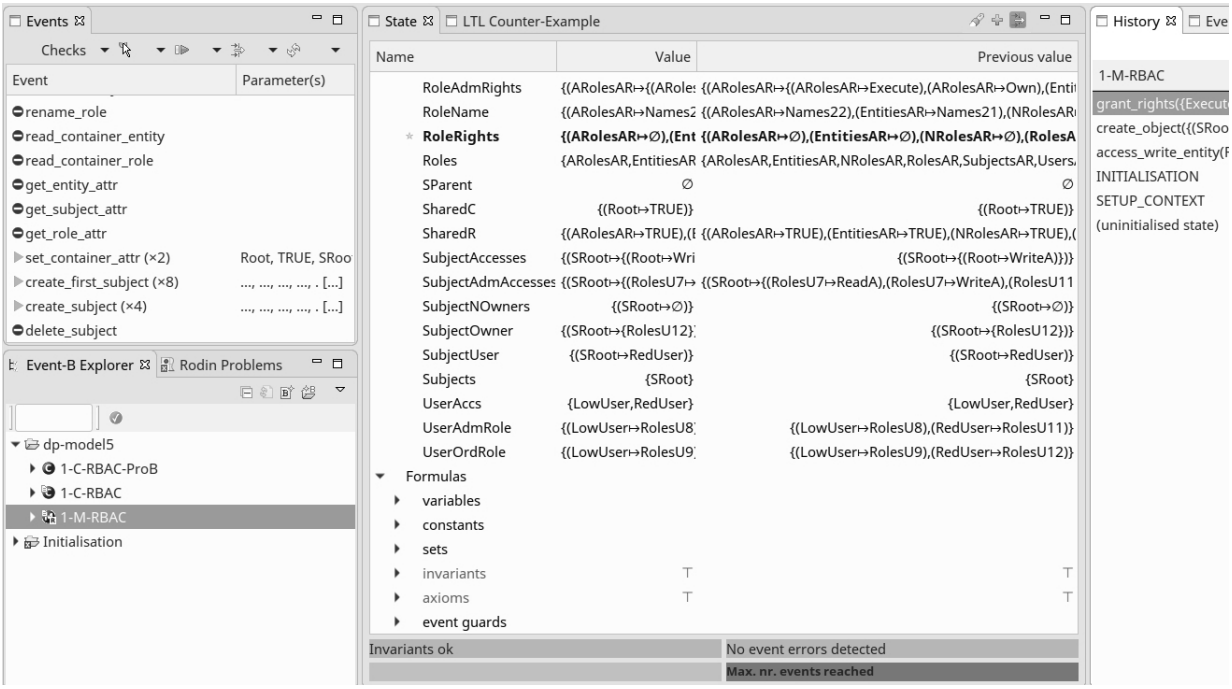

Рис. 3. Возможность выполнения события *create_subject*

Заключение

В настоящей работе на основе опыта использования инструмента проверки моделей ProB для верификации модели управления доступом промышленной ОССН Astra Linux Special Edition (МРОСЛ ДП-модели) в формализованной нотации (на формализованном языке метода Event-B) отмечено, что в большинстве случаев реализованные в ProB алгоритмы перебора значений элементов верифицируемой модели (и, как следствие, наличие проблемы комбинаторного взрыва) требуют доработки описания модели в формализованной нотации. В первую очередь это связано с тем, что некоторые способы представления модели, успешно использовавшиеся при её дедуктивной верификации инструментом Rodin, оказались непригодными для применения инструмента ProB.

Предложенные приёмы по доработке описания МРОСЛ ДП-модели в формализованной нотации, а именно использование нескольких глобальных типов, разделение общих тотальных функций на частные тотальные функции и сокращение числа инвариантов и охранных условий событий, предполагающих перебор подмножеств некоторого множества, создают условия для согласованной верификации описания всех восьми уровней модели инструментом проверки моделей ProB и инструментом дедуктивной верификации Rodin. Эти приёмы также могут быть полезны при разработке

других формальных моделей управления доступом и их верификации с применением соответствующих инструментов.

ЛИТЕРАТУРА

1. `https://astralinux.ru/products/astra-linux-special-edition/` — Операционная система специального назначения Astra Linux Special Edition.
2. *Буренин П. В., Девянин П. Н., Лебеденко Е. В. и др.* Безопасность операционной системы специального назначения Astra Linux Special Edition: учеб. пособие для вузов / под ред. П. Н. Девянина. 3-е изд., перераб. и доп. М.: Горячая линия — Телеком, 2019. 404 с.
3. `https://fstec.ru/component/attachments/download/2832` — Информационное сообщение ФСТЭК России от 15.10.2020 № 240/24/4268.
4. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. 3-е изд., перераб. и доп. М.: Горячая линия — Телеком, 2020. 352 с.
5. *Devyanin P. N., Khoroshilov A. V., Kuliamin V. V., et al.* Integrating RBAC, MIC, and MLS in verified hierarchical security model for operating system // Program. Comput. Soft. 2020. V. 46. P. 443–453.
6. *Abrial J.-R.* Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010.
7. *Abrial J.-R., Butler M., Hallerstede S., et al.* Rodin: An open toolset for modelling and reasoning in Event-B // Intern. J. Software Tools for Technology Transfer. 2010. V. 12. No. 6. P. 447–466.
8. *Девянин П. Н., Ефремов Д. В., Кулямин В. В. и др.* Моделирование и верификация политик безопасности управления доступом в операционных системах. М.: Горячая линия — Телеком, 2019. 214 с.
9. *Девянин П. Н.* Уровень запрещающих ролей иерархического представления МРОСЛ ДП-модели // Прикладная дискретная математика. 2018. № 39. С. 58–71.
10. *Девянин П. Н.* Подходы к моделированию управления доступом в СУБД PostgreSQL в рамках МРОСЛ ДП-модели // Прикладная дискретная математика. Приложение. 2018. № 11. С. 95–98.
11. *Девянин П. Н.* О результатах формирования иерархического представления МРОСЛ ДП-модели // Прикладная дискретная математика. Приложение. 2016. № 9. С. 83–87.
12. *Abrial J.-R. and Hallerstede S.* Refinement, decomposition, and instantiation of discrete models: Application to Event-B // Fundamenta Informaticae. 2007. V. 77. Iss. 1–2. P. 1–28.
13. *Девянин П. Н., Леонова М. А.* Применение подтипов и тотальных функций формального метода Event-B для описания и верификации МРОСЛ ДП-модели // Программная инженерия. 2020. Т. 11. № 4. С. 230–241.
14. *Leuschel M. and Butler M.* ProB: an automated analysis toolset for the B method // Int. J. Softw. Tools Technol. Transf. 2008. No. 10(2). P. 185–203.

REFERENCES

1. `https://astralinux.ru/products/astra-linux-special-edition/` — OS Astra Linux Special Edition, 2020.
2. *Burenin P. V., Devyanin P. N., Lebedenko E. V., et al.* Bezopasnost' operacionnoy sistemy special'nogo naznacheniya Astra Linux Special Edition [Security of Operating System Astra Linux Special Edition]. Moscow, Goryachaya liniya — Telekom, 2019. 404 p. (in Russian)
3. `https://fstec.ru/component/attachments/download/2832` — Information message of FSTEC Russia dated 15.10.2020 no. 240/24/4268.

4. *Devyanin P. N.* Modeli bezopasnosti komp'yuternyh sistem. Upravlenie dostupom i informacionnymi potokami [The Models of Security of Computer Systems: Access Control and Information Flows]. Moscow, Goryachaya liniya — Telekom, 2020. 352 p. (in Russian)
5. *Devyanin P. N., Khoroshilov A. V., Kuliamin V. V., et al.* Integrating RBAC, MIC, and MLS in verified hierarchical security model for operating system. Program. Comput. Soft., 2020, vol. 46, pp. 443–453.
6. *Abrial J.-R.* Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010.
7. *Abrial J.-R., Butler M., Hallerstede S., et al.* Rodin: An open toolset for modelling and reasoning in Event-B. Intern. J. Software Tools for Technology Transfer, 2010, vol. 12, no. 6, pp. 447–466.
8. *Devyanin P. N., Efremov D. V., Kuliamin V. V., et al.* Modelirovanie i verifikaciya politik bezopasnosti upravleniya dostupom v operacionnyh sistemah [Modeling and Verification Access Control Security Policies on Operating Systems]. Moscow, Goryachaya liniya — Telekom, 2019. 214 p. (in Russian)
9. *Devyanin P. N.* Uroven' zapreschayuschih roley ierarhicheskogo predstavleniya MROSL DP-modeli [The level of negative roles of the hierarchical representation of MROSL DP-model]. Prikladnaya Diskretnaya Matematika, 2018, no. 39, pp. 58–71. (in Russian)
10. *Devyanin P. N.* Podhody k modelirovaniyu upravleniya dostupom v SUBD PostgreSQL v ramkah MROSL DP-modeli [Approaches to formal modeling of access control in PostgreSQL within framework of the MROSL DP-model]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2018, no. 11, pp. 95–98. (in Russian).
11. *Devyanin P. N.* O rezul'tatah formirovaniya ierarhicheskogo predstavleniya MROSL DP-modeli [About results of design hierarchical representation of MROSL DP-model]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2016, no. 9, pp. 83–87. (in Russian)
12. *Abrial J.-R. and Hallerstede S.* Refinement, decomposition, and instantiation of discrete models: Application to Event-B. Fundamenta Informaticae, 2007, vol. 77, iss. 1–2, pp. 1–28.
13. *Devyanin P. N. and Leonova M. A.* Primenenie podtipov i total'nyh funkciy formal'nogo metoda Event-B dlya opisaniya i verifikacii MROSL DP-modeli [Application of subtypes and total functions of Event-B formal method for the formalization and verification of the MROSL DP-model]. Programmnaya Ingeneria, 2020, vol. 11, no. 4, pp. 230–241. (in Russian)
14. *Leuschel M. and Butler M.* ProB: an automated analysis toolset for the B method. Int. J. Softw. Tools Technol. Transf., 2008, no. 10(2), pp. 185–203.