ASTRA LINUX®

Приёмы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения её верификации инструментом проверки моделей ProB

Леонова М.А., Девянин П.Н.

ФОРМАЛЬНЫЕ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ И ИХ ВЕРИФИКАЦИЯ

Формальные модели управления доступом:

> Модель Белла-ЛаПадулы – ОС Multics (*Bell D.E., LaPadula L.J.* Secure Computer Systems: Unified Exposition and Multics Interpretation. Bedford, Mass.: MITRE Corp., 1976. MTR-2997 Rev. 1);

> Модель Биба – ОС Multics, ОС Microsoft Windows Vista/7/10 (*Biba K.J.* Integrity Considerations for Secure Computer Systems. Bedford, Mass.: MITRE Corp., 1975. MTR-3153);

…

> МРОСЛ ДП-модель – ОССН Astra Linux Special Edition (Безопасность операционной системы специального назначения Astra Linux Special Edition. Учебное пособие для вузов / П.В. Буренин, П.Н. Девянин, Е.В. Лебеденко и др.; Под ред. доктора техн. наук, профессора П.Н. Девянина. 3-е издание, перераб. и доп. М.: Горячая линия – Телеком, 2019. 404 с.: ил.).

Верификация формальных моделей управления доступом:

> Event-B, Rodin (*Abrial J.-R., Butler M., Hallerstede S. et al.* Rodin: An Open Toolset for Modelling and Reasoning in Event-B // International Journal on Software Tools for Technology Transfer. 2010. Vol. 12, no. 6. P. 447–466);

…

> AstraVer Toolset (*Девянин П.Н., Ефремов Д.В., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В.* Моделирование и верификация политик безопасности управления доступом в операционных системах. М.: Горячая линия – Телеком, 2019. 214 с.: ил.).

ТРЕБОВАНИЯ ПО БЕЗОПАСНОСТИ ИНФОРМАЦИИ, УСТАНАВЛИВАЮЩИЕ 6 УРОВНЕЙ ДОВЕРИЯ (ПРИКАЗ ФСТЭК РОССИИ ОТ 30.07.2018 № 131)

5 УРОВЕНЬ ДОВЕРИЯ (ОБЪЕКТЫ КИИ 2 КАТЕГОРИИ, ГИС 2 КЛАССА ЗАЩИЩЕННОСТИ)

- идентификация и анализ скрытых каналов по памяти.

4 УРОВЕНЬ ДОВЕРИЯ (ОБЪЕКТЫ КИИ 1 КАТЕГОРИИ, ГИС 1 КЛАССА ЗАЩИЩЕННОСТИ)

- модель безопасности, включая реализуемые политики управления доступом и фильтрации информационных потоков.

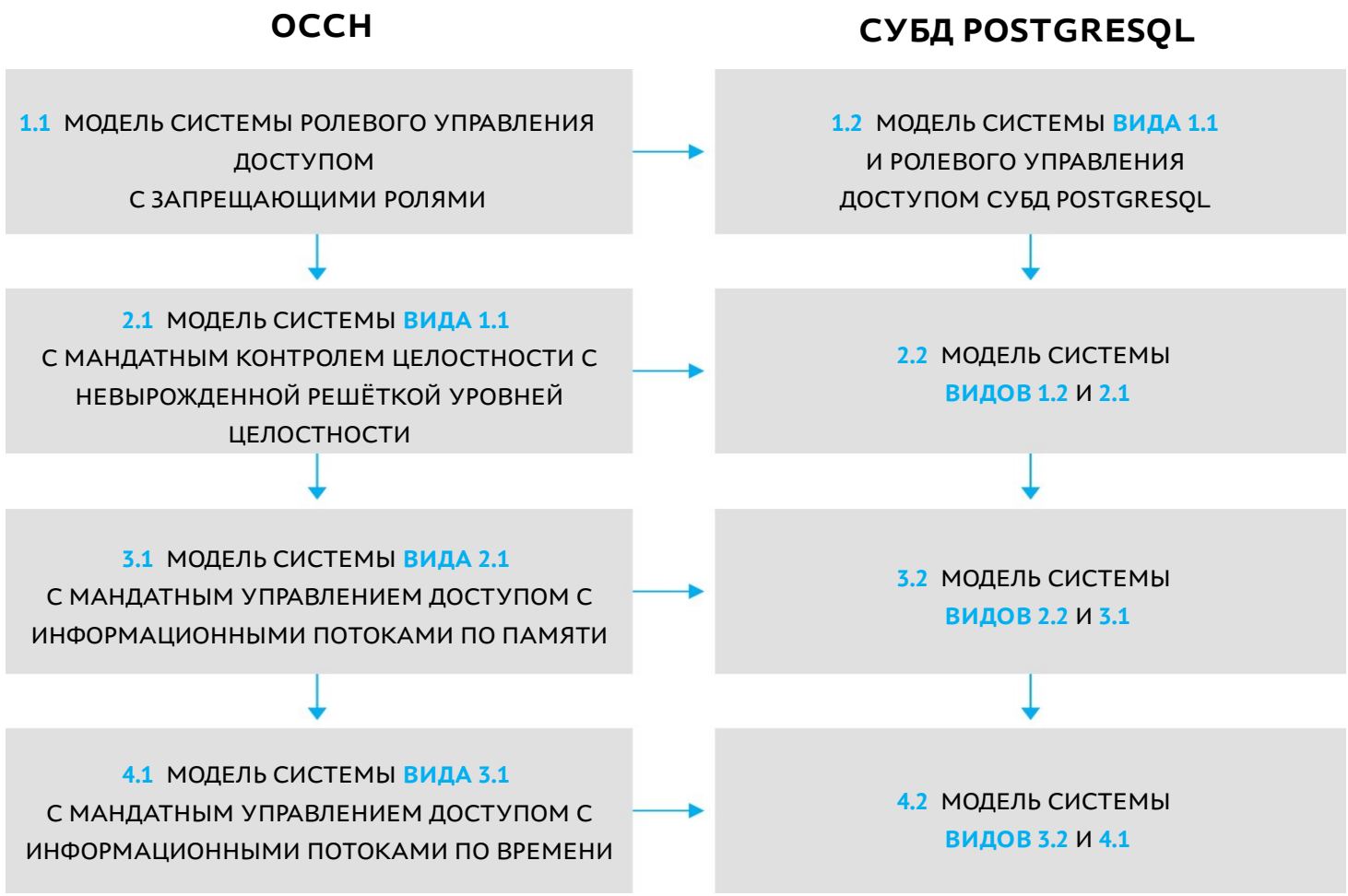
3 УРОВЕНЬ ДОВЕРИЯ (ИС, В КОТОРЫХ ОБРАБАТЫВАЕТСЯ ИНФОРМАЦИЯ, СОДЕРЖАЩАЯ СЕКРЕТНЫЕ СВЕДЕНИЯ)

- верификация модели безопасности с использованием инструментальных средств;
- идентификация и анализ скрытых каналов по времени.

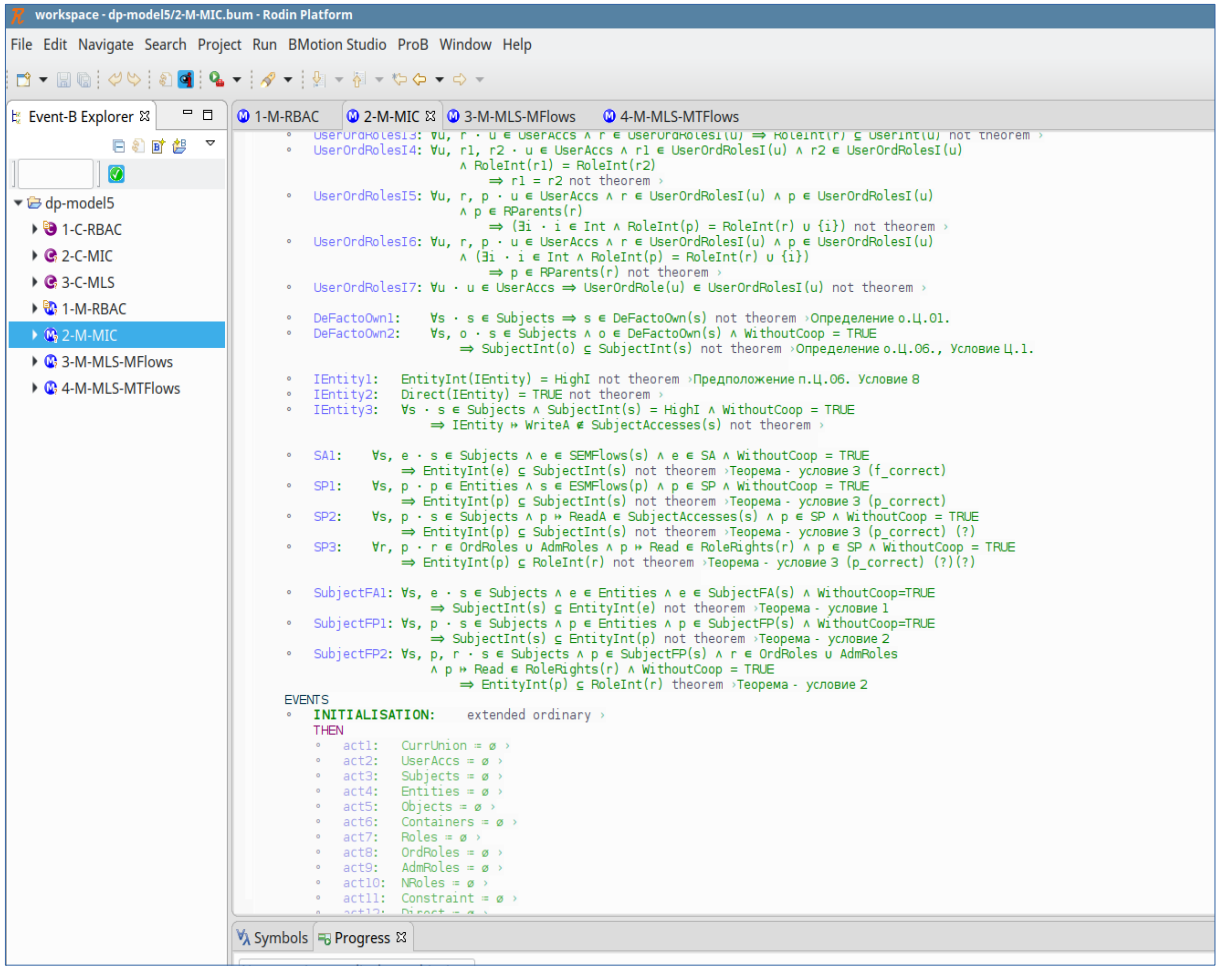
1 УРОВЕНЬ ДОВЕРИЯ (ИС, В КОТОРЫХ ОБРАБАТЫВАЕТСЯ ИНФОРМАЦИЯ, СОДЕРЖАЩАЯ СВЕДЕНИЯ ОСОБОЙ ВАЖНОСТИ)

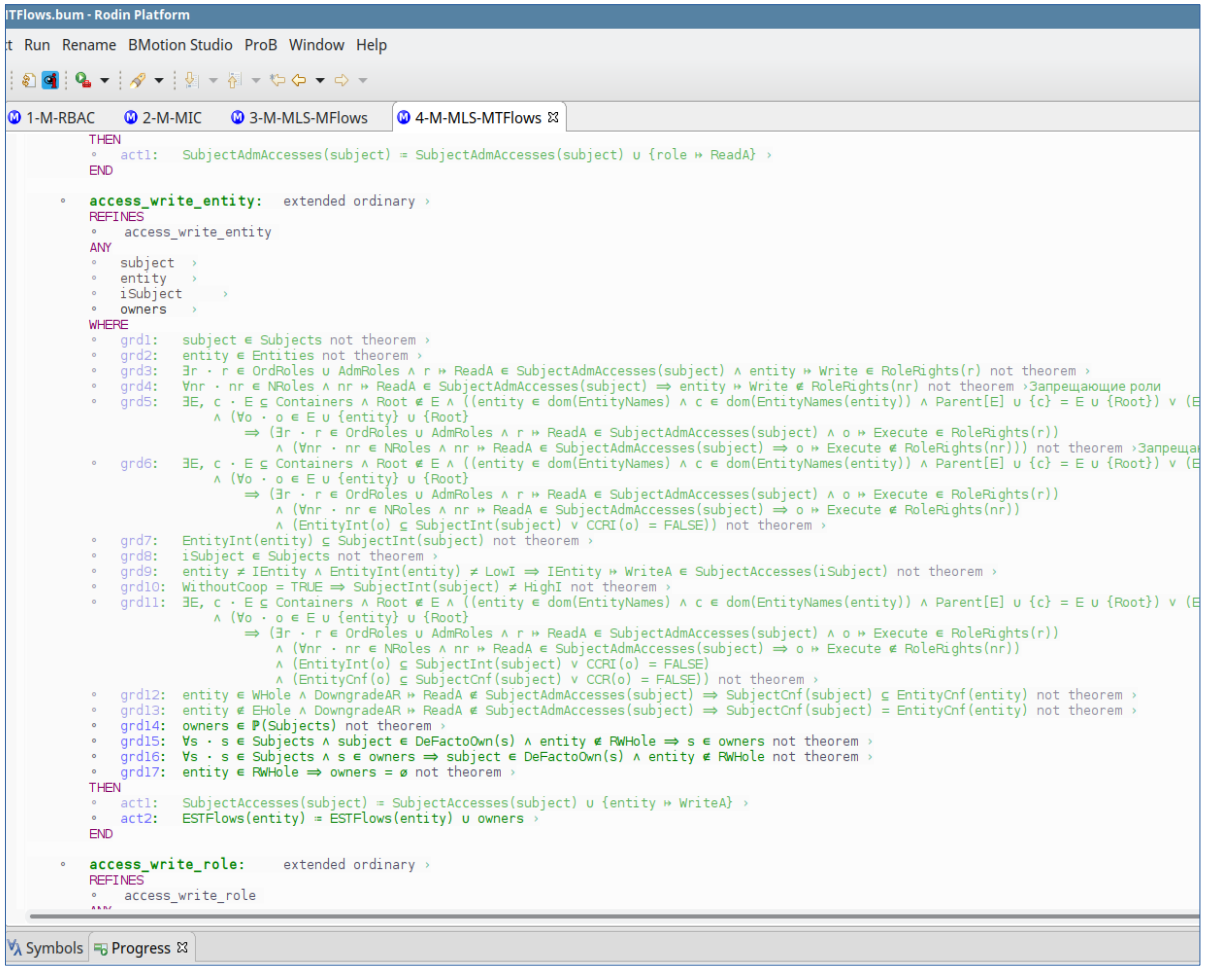
- идентификация и анализ скрытых статистических каналов.

ФОРМИРОВАНИЕ ИЕРАРХИЧЕСКОГО ПРЕДСТАВЛЕНИЯ МРОСЛ ДП-МОДЕЛИ В МАТЕМАТИЧЕСКОЙ НОТАЦИИ



ПРИМЕР ОПИСАНИЯ МРОСЛ ДП-МОДЕЛИ В ФОРМАЛИЗОВАННОЙ НОТАЦИИ





РЕШАЕМЫЕ ЗАДАЧИ ИССЛЕДОВАНИЯ

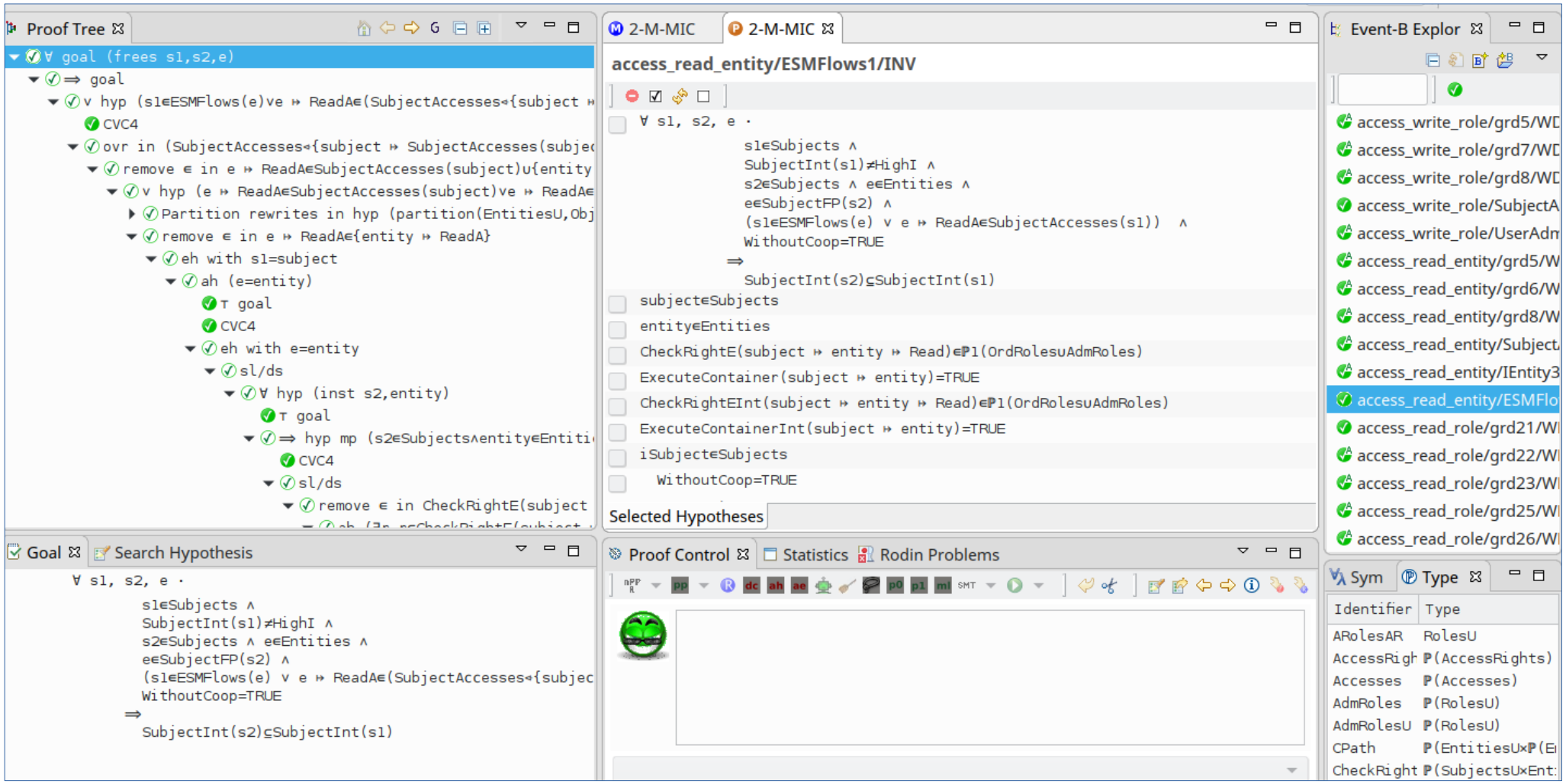
1. Актуализация МРОСЛ ДП-модели в формализованной нотации в соответствие с изменениями ее описания в математической нотации, как для ОССН, так и для СУБД PostgreSQL.

2. Автоматизированная дедуктивная верификация с применением инструментального средства Rodin.

3. Нахождение приемов формирования МРОСЛ ДП-модели в математической и формализованной нотациях для их приближения к практической реализации механизма управления доступом в ОССН и упрощения дедуктивной верификации с применением инструментального средства Rodin.

4. Исследование возможности верификации МРОСЛ ДП-модели и реализации автоматизированных тестов механизма управления доступом ОССН с использованием инструмента проверки моделей ProB.

ВЕРИФИКАЦИЯ 2 УРОВНЯ (МКЦ) МРОСЛ ДП-МОДЕЛИ НА ЯЗЫКЕ МЕТОДА EVENT-B ИНСТРУМЕНТАЛЬНЫМ СРЕДСТВОМ RODIN

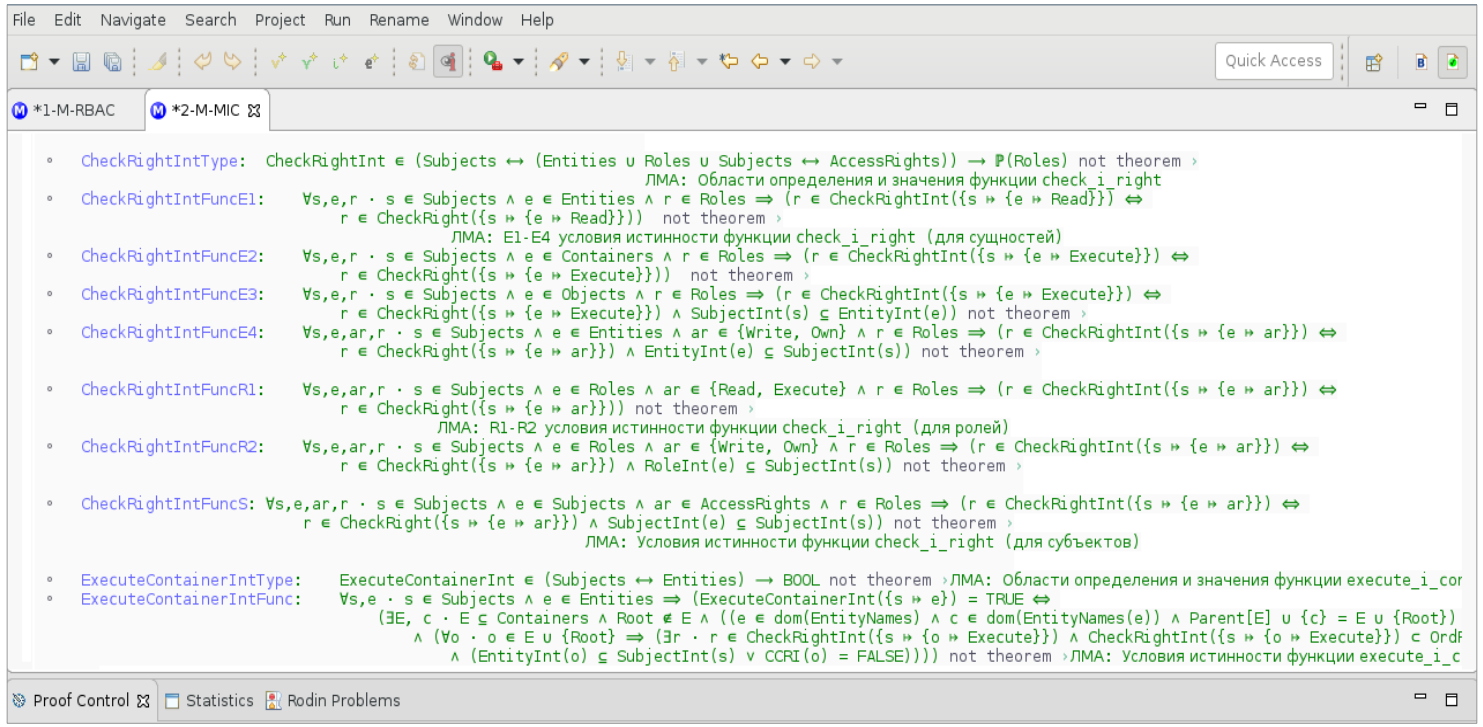


ПРИМЕР ИСПОЛЬЗОВАНИЯ ТОТАЛЬНЫХ ФУНКЦИЙ


```
int pdpml_permission(
      const PDPML_T *s, // контекст процесса (субъект-сессии)
      const PDPML_T *o, // контекст файла (сущности)
      int mode          // запрашиваемые доступы
)
{
      ...
      // Проверка условий мандатного управления доступом
      if (pdpml_conf_permission(s, o, mode) ) {
            return -EPERM;
      }
      // Проверка условий мандатного контроля целостности
      if (pdp_ilev_permission(s->ilev, o->ilev, mode) ) {
            return -EPERM;
      }
      return 0;
}
```

Фрагмент программного кода функции pdpml_permission в ОССН

Задание тотальных функций CheckRight и ExecuteContainer в формализованной нотации



ПРИМЕР ИСПОЛЬЗОВАНИЯ ТОТАЛЬНЫХ ФУНКЦИЙ


```
access_read_entity:    extended ordinary ›
REFINES
  access_read_entity
ANY
  subject   ›
  entity    ›
WHERE
  grd1:   subject ∈ Subjects not theorem ›
  grd2:   entity ∈ Entities not theorem ›
  grd3:   ∃r · r ∈ OrdRoles ∪ AdmRoles ∧ r ↦ ReadA ∈ SubjectAdmAccesses(subject) ∧ entity ↦ Read ∈ RoleRights
  grd4:   ∀nr · nr ∈ NRoles ∧ nr ↦ ReadA ∈ SubjectAdmAccesses(subject) ⇒ entity ↦ Read ∉ RoleRights(nr) not
  grd5:   ∃E, c · E ⊆ Containers ∧ Root ∉ E ∧ ((entity ∈ dom(EntityNames) ∧ c ∈ dom(EntityNames(entity)) ∧ Pa
          ∧ (∀o · o ∈ E ∪ {entity} ∪ {Root}
              ⇒ (∃r · r ∈ OrdRoles ∪ AdmRoles ∧ r ↦ ReadA ∈ SubjectAdmAccesses(subject) ∧ o ↦ Execute ∈
                  ∧ (∀nr · nr ∈ NRoles ∧ nr ↦ ReadA ∈ SubjectAdmAccesses(subject) ⇒ o ↦ Execute ∉ RoleRi
  grd6:   ∃E, c · E ⊆ Containers ∧ Root ∉ E ∧ ((entity ∈ dom(EntityNames) ∧ c ∈ dom(EntityNames(entity)) ∧ Pa
          ∧ (∀o · o ∈ E ∪ {entity} ∪ {Root}
              ⇒ (∃r · r ∈ OrdRoles ∪ AdmRoles ∧ r ↦ ReadA ∈ SubjectAdmAccesses(subject) ∧ o ↦ Execute ∈
                  ∧ (∀nr · nr ∈ NRoles ∧ nr ↦ ReadA ∈ SubjectAdmAccesses(subject) ⇒ o ↦ Execute ∉ RoleRi
                  ∧ (EntityInt(o) ⊆ SubjectInt(subject) ∨ CCRI(o) = FALSE)) not theorem ›
  grd7:   WithoutCoop = TRUE ⇒ SubjectInt(subject) ≠ HighI not theorem ›
THEN
  act1:   SubjectAccesses(subject) ≔ SubjectAccesses(subject) ∪ {entity ↦ ReadA} ›
END
```

Реализация функций модели в математической нотации в виде охранных условий в формализованной нотации

Реализация функций в математической нотации в виде тотальных функций в формализованной нотации

```
access_read_entity:    extended ordinary ›
REFINES
  access_read_entity
ANY
  subject   ›
  entity    ›
  iSubject      ›ЛМА
WHERE
  grd1:   subject ∈ Subjects not theorem ›
  grd2:   entity ∈ Entities not theorem ›
  grd3:   CheckRightE(subject ↦ entity ↦ Read) ∈ ℙ1(OrdRoles ∪ AdmRoles) not theorem ›ЛМА: Замена у
  grd4:   ExecuteContainer(subject ↦ entity) = TRUE not theorem ›ЛМА: Замена условий значением функ
  grd5:   CheckRightEInt(subject ↦ entity ↦ Read) ∈ ℙ1(OrdRoles ∪ AdmRoles) not theorem ›ЛМА: Заме
  grd6:   ExecuteContainerInt(subject ↦ entity) = TRUE not theorem ›ЛМА: Замена условий значением ф
  grd7:   iSubject ∈ Subjects not theorem ›ЛМА: Добавлен в соответствии с моделью
  grd8:   WithoutCoop = TRUE ⇒ SubjectInt(subject) ≠ HighI not theorem ›
THEN
  act1:   SubjectAccesses(subject) ≔ SubjectAccesses(subject) ∪ {entity ↦ ReadA} ›
END
```

ИСПОЛЬЗОВАНИЕ НЕСКОЛЬКИХ ГЛОБАЛЬНЫХ ТИПОВ С СОХРАНЕНИЕМ ПОДХОДА ПО ПРИМЕНЕНИЮ ПОДТИПОВ


```
SETS
  UsersU    ›
  SubjectsU    ›
  EntitiesU    ›
  RolesU    ›
  Names    ›NAMES: множество возможных имён сущностей, ролей, административных ролей
  Accesses    ›Ra: множество видов доступа
  AccessRights    ›Rr: множество видов прав доступа
```

Глобальные типы в формализованной нотации

Подтипы в формализованной нотации

```
EntitiesUPartition: partition(EntitiesU, ObjectsU, ContainersU) not theorem ›
RolesUPartition:    partition(RolesU, AdmRolesU, OrdRolesU, NRolesU) not theorem ›
```

```
CheckRightEType:    CheckRightE ∈ Subjects × Entities × AccessRights ⟶ ℙ(Roles) not theorem ›ЛМА: Области определения и значения функции check_right
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)) not theorem ›ЛМА: Условия истинности функции check_right (для сущностей)
CheckRightRType:    CheckRightR ∈ Subjects × Roles × AccessRights ⟶ ℙ(AdmRoles) not theorem ›
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)) not theorem ›ЛМА: Условия истинности функции check_right (для ролей)
CheckRightSType:    CheckRightS ∈ Subjects × Subjects × {Own} ⟶ ℙ(Roles) not theorem ›
CheckRightSFunc1:   ∀s,e,r · s ∈ Subjects ∧ e ∈ Subjects ∧ r ∈ OrdRoles ∪ AdmRoles ⇒ (r ∈ CheckRightS(s ↦ e ↦ Own) ⇔
                        r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧ r ∈ SubjectOwner(e)) not theorem ›ЛМА: S1-S2 условия истинности функции check_right (для субъекто
CheckRightSFunc2:   ∀s,e,r · s ∈ Subjects ∧ e ∈ Subjects ∧ r ∈ NRoles ⇒ (r ∈ CheckRightS(s ↦ e ↦ Own) ⇔
                        r ↦ ReadA ∈ SubjectAdmAccesses(s) ∧ r ∈ SubjectNOwners(e)) not theorem ›
```

ЗАДАНИЕ ОХРАННЫХ УСЛОВИЙ СОБЫТИЙ С УЧЕТОМ СПЕЦИФИКИ PROB


```
◦  grd9:   ∀u · u ∈ Union ∧ admRole ∈ UserRoles(u) ⇒ role ∉ UserRoles(u) not theorem ›
◦  grd10:  ∀u · u ∈ Union ∧ admRole ∈ UserRoles(u) ⇒ role ∉ CommonRoles not theorem ›
◦  grd11:  admRights ⊆ RoleAdmRights(admRole) not theorem ›
◦  grd12:  ∀ar · ar ∈ accessRights ⇒ role ↦ ar ∈ admRights not theorem ›
◦  grd13:  ∀ar · ar ∉ accessRights ⇒ role ↦ ar ∉ admRights not theorem ›
◦  grd14:  Read ∉ accessRights ⇒ (∀r · r ∈ Roles \ {role} ⇒ r ∉ dom(admRights)) not theorem ›
◦  grd15:  ∀r, ar · r ∈ Roles ∧ r ≠ role ∧ ar ∈ AccessRights ∧ ar ≠ Read ⇒ r ↦ ar ∉ admRights not theorem ›
◦  grd16:  ∀r, p · r ∈ Roles ∧ p ∈ RParents(r) ∧ r ↦ Read ∈ admRights ⇒ p ↦ Read ∈ admRights not theorem ›
◦  grd17:  Read ∈ accessRights
              ⇒ (∀u, r · u ∈ UserAccs ∧ admRole ∈ UserRoles(u) ∧ r ∈ dom(admRights)
                   ⇒ r ∉ Constraint(UserAdmRole(u)) ∪ Constraint(UserOrdRole(u)) ∪ Constraint(CommonRole)) not theorem ›Запрещающие рол
◦  grd18:  depth ∈ ℕ ⟶ ℙ(dom(admRights)) not theorem ›
◦  grd19:  ∀r · r ∈ dom(admRights) ⇒ (∃i · i ∈ ℕ ∧ r ∈ depth(i)) not theorem ›
◦  grd20:  depth(0) = {role} not theorem ›
◦  grd21:  ∀i · i ∈ ℕ ⇒ (∀p · p ∈ depth(i + 1) ⇒ (∃ch · ch ∈ depth(i) ∧ p ∈ RParents(ch))) not theorem ›
◦  grd22:  ∀i · i ∈ ℕ ∧ (∀r · r ∈ depth(i) ⇒ (∀u · u ∈ Union ∧ admRole ∈ UserRoles(u) ⇒ r ∉ UserRoles(u) ∧ r ∉ CommonRoles))
              ⇒ (∀r · r ∈ depth(i + 1) ⇒ (∀u · u ∈ Union ∧ admRole ∈ UserRoles(u) ⇒ r ∉ UserRoles(u) ∧ r ∉ CommonRoles)) theorem ›
◦  grd23:  ∀i · i ∈ ℕ ⇒ (∀r · r ∈ depth(i) ⇒ (∀u · u ∈ Union ∧ admRole ∈ UserRoles(u) ⇒ r ∉ UserRoles(u) ∧ r ∉ CommonRoles)) theorem ›
THEN
◦  act1:   RoleAdmRights(admRole) ≔ RoleAdmRights(admRole) \ admRights ›
END
```

Использование индукции в событии

Функция потомков ролей

```
DescendantsR1:  ∀r,d · r ∈ Roles ∧ d ∈ Roles ⇒ (d ∈ DescendantsR(r) ⇔ d = r ∨ (d ≠ r ∧ (∃k · k ∈ KidsR(r) ∧ d ∈ DescendantsR(k))))
UserRoles1: ∀u · u ∈ UsersU ⇒ UserRoles(u) ⊆ AdmRoles ∪ OrdRoles not theorem ›
UserRoles2: ∀u, r · u ∈ UsersU ∧ r ∈ UserRoles(u) ⇒ DescendantsR(r) ⊆ UserRoles(u) not theorem ›
UserRoles4: ∀u · u ∈ UserAccs ⇒ UserOrdRole(u) ∈ UserRoles(u) not theorem ›
UserRoles5: ∀u · u ∈ UserAccs ⇒ UserAdmRole(u) ∈ UserRoles(u) not theorem ›
UserRoles6: ∀u, r · u ∈ UsersU ∧ r ∈ Roles \ UserRoles(u) ⇒ DescendantsR(r) ∩ UserRoles(u) = ∅ not theorem ›
```

Использование функции потомков в событии

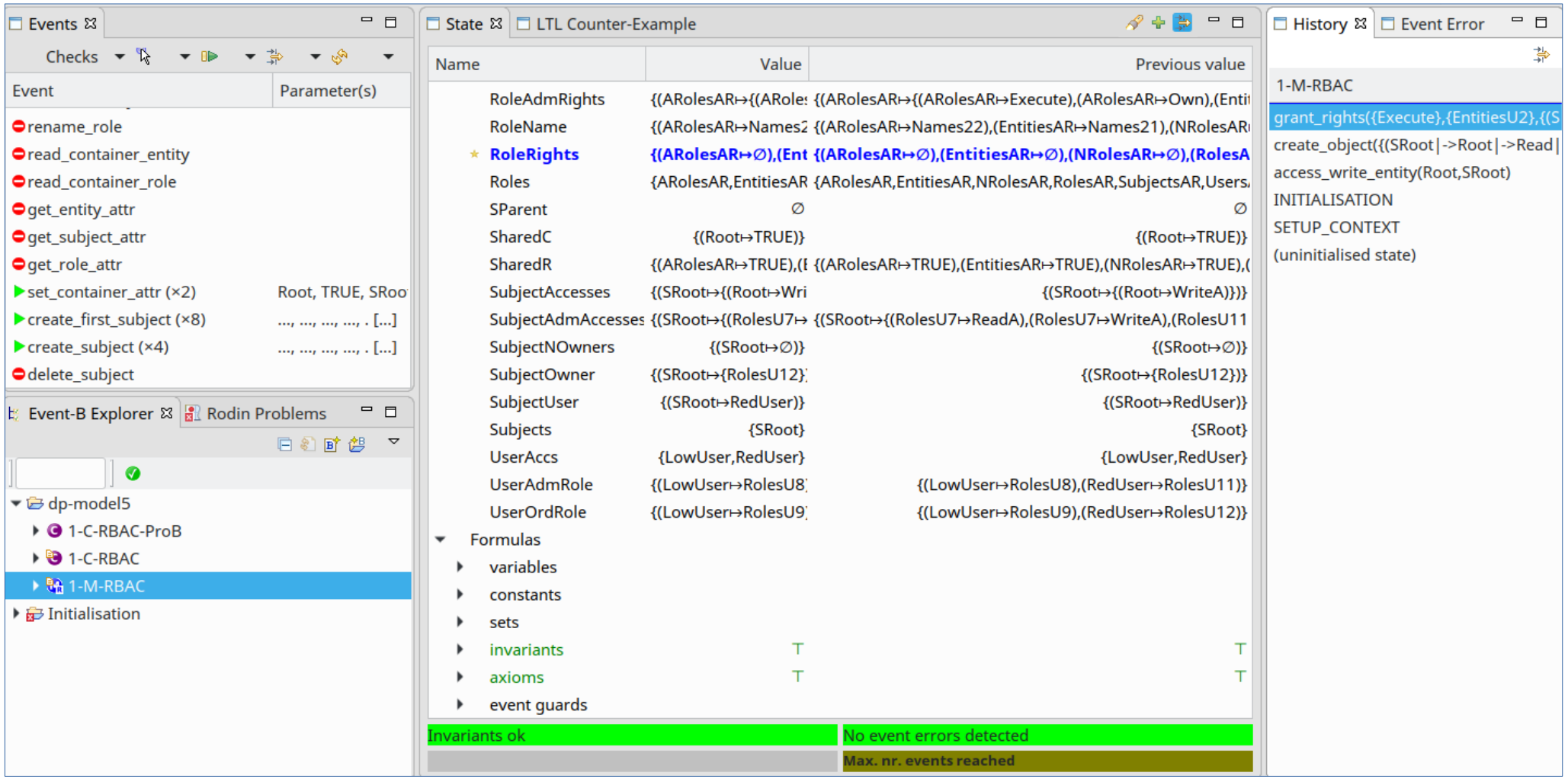
```
◦  grd9:   ∀u · u ∈ UsersU ∧ admRole ∈ UserRoles(u) ⇒ role ∉ UserRoles(u) not theorem ›
◦  grd10:  ∀u · u ∈ UsersU ∧ admRole ∈ UserRoles(u) ⇒ role ∉ CommonRoles not theorem ›
◦  grd11:  admRights ∈ {r | r ∈ Roles ∧ role ∈ DescendantsR(r)} ↔ accessRights not theorem ›
◦  grd12:  accessRights = {Read} ⇒ admRights = {r | r ∈ Roles ∧ role ∈ DescendantsR(r)} × {Read} not theorem ›
◦  grd13:  accessRights = {Write} ⇒ admRights = {role ↦ Write} not theorem ›
◦  grd14:  accessRights = {Read, Write} ⇒ admRights = ({r | r ∈ Roles ∧ role ∈ DescendantsR(r)} × {Read}) ∪ {role ↦ Write} not theorem ›
◦  grd15:  Read ∈ accessRights
              ⇒ (∀u, r · u ∈ UserAccs ∧ admRole = UserAdmRole(u) ∧ r ∈ NRoles ∧ r ∈ dom(admRights)
                   ⇒ r ∉ Constraint(UserAdmRole(u)) ∪ Constraint(UserOrdRole(u)) ∪ Constraint(CommonRole)) not theorem ›
◦  grd16:  checkRightR ∈ Subjects × Roles × AccessRights ⟶ ℙ(AdmRoles) not theorem ›ЛМА: grd 16-19 условия переопределения функции check_right
◦  grd17:  ∀s,e,ar · s ∈ Subjects ∧ e ∈ Roles ∧ ar ∉ accessRights ⇒ checkRightR(s ↦ e ↦ ar) = CheckRightR(s ↦ e ↦ ar) not theorem ›
◦  grd18:  ∀s,e,ar,r · s ∈ Subjects ∧ e ∈ Roles ∧ ar ∈ accessRights ∧ r ∈ AdmRoles \ {admRole} ⇒ (r ∈ checkRightR(s ↦ e ↦ ar) ⇔ r ∈ CheckRightR(s ↦ e ↦ ar))
◦  grd19:  ∀s,e,ar · s ∈ Subjects ∧ e ∈ Roles ∧ ar ∈ accessRights ⇒ (admRole ∈ checkRightR(s ↦ e ↦ ar) ⇔
              admRole ↦ ReadA ∈ SubjectAdmAccesses(s) ∧ e ↦ ar ∈ RoleAdmRights(admRole) \ admRights) not theorem ›
THEN
◦  act1:   RoleAdmRights(admRole) ≔ RoleAdmRights(admRole) \ admRights ›
◦  act2:   CheckRightR ≔ checkRightR ›ЛМА: Переопределение функции check_right
END
```

ИНИЦИАЛИЗАЦИЯ НАЧАЛЬНОГО СОСТОЯНИЯ МОДЕЛИ ДЛЯ ИСПОЛЬЗОВАНИЯ PROB


```
EVENTS
  INITIALISATION:    not extended ordinary ›
  THEN
    act1:   UserAccs ≔ {RedUser, LowUser} ›
    act2:   Subjects ≔ {SRoot} ›
    act3:   Entities ≔ {Root} ›
    act4:   Objects ≔ ∅ ›
    act5:   Containers ≔ {Root} ›
    act6:   Roles ≔ PredefinedRoles ›
    act7:   OrdRoles ≔ CommonRoles ∪ {RUOrdRole, LUOrdRole} ∪ HR ›
    act8:   AdmRoles ≔ SpecialAdmRoles ∪ {RUAdmRoleH, RUAdmRoleL, LUAdmRole} ›
    act9:   NRoles ≔ ∅ ›
    act10:  Constraint ≔ PredefinedRoles × {∅} ›
    act11:  DirectE ≔ {Root} × {TRUE} ›
    act12:  DirectR ≔ PredefinedRoles × {TRUE} ›
    act13:  EntityMP ≔ {Root ↦ Root} ›
    act14:  EntityNames ≔ ∅ ›
    act15:  Parent ≔ ∅ ›
    act16:  CPath ≔ {Root ↦ {Root}} ›
    act17:  KidsR ≔ ((PredefinedRoles \ HR) × {∅}) ∪ {Role1 ↦ {Role2, Role3, Role4, Role5}, Role2 ↦ {Role5}, Role3 ↦ {Role5}, Role4 ↦ {Role6, Role8},
                Role5 ↦ {Role7}, Role6 ↦ {Role7, Role8}, Role7 ↦ {Role9}, Role8 ↦ {Role9}, Role9 ↦ ∅} ›
    act18:  DescendantsR ≔ {r ↦ {r} | r ∈ PredefinedRoles \ HR} ∪ {Role1 ↦ HR, Role2 ↦ {Role2, Role5, Role7, Role9}, Role3 ↦ {Role3, Role5, Role7, Role9},
                Role4 ↦ {Role4, Role6, Role7, Role8, Role9}, Role5 ↦ {Role5, Role7, Role9}, Role6 ↦ {Role6, Role7, Role8, Role9}, Role7 ↦ {Role7, Role9},
                Role8 ↦ {Role8, Role9}, Role9 ↦ {Role9}} ›
    act19:  RoleAdmRights ≔ {
                RUAdmRoleL ↦ (PredefinedRoles × {Execute}) ∪ (({RUAdmRoleL, RUOrdRole, CommonRole} ∪ SpecialAdmRoles) × {Read, Write}),
                RUAdmRoleH ↦ (PredefinedRoles × {Execute}),
                LUAdmRole ↦ (PredefinedRoles × {Execute}) ∪ ({LUAdmRole, LUOrdRole, CommonRole} × {Read, Write}),
                RolesAR ↦ {x ↦ y | x ∈ PredefinedRoles ∧ y = Execute} ∪ {x ↦ y | x ∈ PredefinedRoles ∩ OrdRolesU ∧ y = Own},
                ARolesAR ↦ {x ↦ y | x ∈ PredefinedRoles ∧ y = Execute} ∪ {x ↦ y | x ∈ PredefinedRoles ∩ AdmRolesU ∧ y = Own},
                NRolesAR ↦ {x ↦ y | x ∈ PredefinedRoles ∧ y = Execute} ∪ {x ↦ y | x ∈ PredefinedRoles ∩ NRolesU ∧ y = Own}}
                ∪ {ri ↦ RE | ri ∈ SpecialAdmRoles \ {RolesAR, ARolesAR, NRolesAR} ∧ RE = PredefinedRoles × {Execute}} ›
    act20:  RoleRights ≔ ((PredefinedRoles \ {RUOrdRole}) × {∅}) ∪ {RUOrdRole ↦ {Root} × AccessRights} ›
    act21:  RoleName :∈ PredefinedRoles ⤖ Names ›
    act22:  SubjectAccesses ≔ {SRoot ↦ ∅} ›
    act23:  SubjectAdmAccesses ≔ {SRoot ↦ {RUAdmRoleL, RUOrdRole, CommonRole} × Accesses} ›
    act24:  SubjectNOwners ≔ {SRoot ↦ ∅} ›
    act25:  SubjectOwner ≔ {SRoot ↦ {RUOrdRole}} ›
    act26:  SubjectUser ≔ {SRoot ↦ RedUser} ›
    act27:  SharedC ≔ {Root} × {TRUE} ›
    act28:  SharedR ≔ PredefinedRoles × {TRUE} ›
    act29:  SParent ≔ ∅ ›
    act30:  UserOrdRole ≔ {RedUser ↦ RUOrdRole, LowUser ↦ LUOrdRole} ›
    act31:  UserAdmRole ≔ {RedUser ↦ RUAdmRoleL, LowUser ↦ LUAdmRole} ›
    act32:  CheckRightE ≔ {SRoot} × {Root} × AccessRights × {{RUOrdRole}} ›
    act33:  CheckRightR ≔ {a ↦ b ↦ c ↦ {d} | a = SRoot ∧ b ∈ {RUAdmRoleL} ∪ {RUOrdRole} ∪ {CommonRole} ∪ SpecialAdmRoles ∧ c ∈ {Read, Write} ∧ d = RUAdmRoleL} ∪
                {a ↦ b ↦ c ↦ {d} | a = SRoot ∧ b ∈ PredefinedRoles ∧ c = Execute ∧ d = RUAdmRoleL} ∪
                ((({SRoot} × PredefinedRoles × AccessRights) \
                ({a ↦ b ↦ c | a = SRoot ∧ b ∈ {RUAdmRoleL} ∪ {RUOrdRole} ∪ {CommonRole} ∪ SpecialAdmRoles ∧ c ∈ {Read, Write}} ∪
                {a ↦ b ↦ c | a = SRoot ∧ b ∈ PredefinedRoles ∧ c = Execute})) × {∅}) ›
    act34:  CheckRightS ≔ {SRoot ↦ SRoot ↦ Own ↦ {RUOrdRole}} ›
    act35:  ExecuteContainer ≔ {SRoot ↦ Root ↦ TRUE} ›ЛМА
  END
```

- Учетные записи пользователей, процессы, файлы, каталоги
- Роли (обычные, административные, запрещающие)
- Метки файлов и каталогов, их иерархия
- Иерархия ролей
- Административные и обычные права доступа ролей
- Роли процессов
- Роли учетных записей пользователей
- Права доступа процессов к файлам, каталогам, ролям и процессам
- Права на выполнение к цепочке каталогов в их иерархии

ПРИМЕР ТЕСТИРОВАНИЯ НА СОХРАНЕНИЕ ИНВАРИАНТОВ ПРИ ВЫПОЛНЕНИИ СОБЫТИЙ МОДЕЛИ С ИСПОЛЬЗОВАНИЕМ PROB



ASTRA LINUX®

Спасибо за внимание!