МЕТОД ПРЕДСТАВЛЕННЯ ПРАВИЛ ВИКОРИСТАННЯ ЕКІПАЖІВ В ПІДСИСТЕМІ "АУДИТ" АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ЛЬОТНОЮ ДІЯЛЬНІСТЮ АВІАКОМПАНІЇ

А.В. Мірошниченко, В.О. Барило, О.В. Сабокар
Інститут проблем математичних машин і систем НАН України
E-mail - d_one@inbox.ru

При плануванні льотної діяльності та при прийнятті диспетчерських рішень щодо використання екіпажів авіакомпанії, як правило, створюються плани польотів льотного персоналу, які представляють собою послідовність льотних тижнів (ротацій), вихідних та тренувань екіпажів. В свою чергу, льотний тиждень складається з чередувань чергувань, впродовж яких виконуються польоти та відпочинки екіпажів.

З метою забезпечення безпеки польотів міжнародні та національні авіаційні організації виробили ряд вимог до порядку використання екіпажів. Це, перш за все, торкається часу відпочинку екіпажів в залежності від параметрів періодів чергування. Правила використання екіпажів можуть бути міжнародними, національними або правилами авіакомпанії, вони можуть бути обов'язковими для виконання або необов'язковими, вони можуть брати до уваги різні умови польоту, наприклад, час вступу на чергування, кількість змін часових поясів за час чергування, кількість взльотів/посадок за час чергування, тощо. При плануванні льотної діяльності сучасна авіакомпанія використовує більш як сотню правил. Динаміка появи нових правил або відміни застарілих правил досить велика, що не дає можливості зводити процес планування до розв'язання задач математичного програмування.

Така схема вимагає виділення в якості самостійної компоненти архітектури систем керування екіпажами - підсистеми "Аудит", в функції якої входить ведення бази знань правил використання екіпажів та перевірка на коректність претендентів на рішення. Така підсистема може використовуватися як в плануванні, так і в диспетчерській діяльності. Нижче приведемо ряд положень, які раціонально використовувати при побудові таких систем.

1 Синтаксис та семантика мови правил

Центральною концепцією для системи аудиту є ротація. Уся система правил - це є твердження про те, які значення певні елементи ротації можуть чи не можуть приймати. Якщо змінюється сама ротація (додано новий рейс, посадка у незапланованому місці), то з'являється загроза того, що не всі елементи ротації тепер будуть задовольняти попередньо встановленим нормам польоту (не будуть відповідати правилам).

Проблема представлення правил і перевірки ротації створило проблему конструювання логічної системи. Для розробки такої системи будемо використовувати логіку предикатів, як основу всього набору. При цьому, ускладнюючи сам процес розробки проекту, проте додаючи великих можливостей самому результату, необхідно створити предикатне числення, що може використовувати функції. Зазначимо, що створюється логічна система з використанням функцій, а не функціональна система з логічними конструкціями.

1.1 Предикат (Predicate)

Предикат - це елементарна формула нашої мови. Це неділимий елемент. Єдине його призначення - виразити твердження, яке може бути або правдивим, або хибним. Правдивість чи хибність предиката в системі правил залежить від запису самої формули. Атомарність цього елементу говорить про те, що не має ще меншого виразу, який міг би бути оцінений як правда чи неправда.

Синтаксис наступний:

<predicate>:: <operand-1> <operator> <operand-2>

Операнд - це певна величина чи функція, що повертає певне значення, яке буде порівняне оператором. Перший операнд - це завжди функція. Вона визначає певний параметр ротації і обраховане значення цього параметра або множина значень мають бути порівняні з

"правильними" варіантами. Другий операнд має бути або константою, або множиною констант, або функцією, що також повертає або одне значення, або множину величин.

Оператори порівняння двох незалежних величин можуть використовуватися для порівняння чисел, строк, часових інтервалів та ін. До них входять:

=	два операнда є рівними величинами
!=	два операнда не є рівними величинами
>	перший операнд строго більший за другий
≥	перший операнд рівний другому або більший за нього
<	перший операнд строго менший за другий;
≤	перший операнд рівний другому або менший за нього

Оператори другої групи дозволяють перевірити належність значення, що представляє перший операнд, до множини значень, представленої другим операндом. Ця група операторів включає один оператор для невпорядкованої множини:

IN	значення певної величини, представленої першим операндом належить до множини значень, представленої другим операндом
BETWEEN	значення певної величини, представленої першим операндом є більшим або рівним першому елементу діапазону, представленого другим операндом, та меншим або рівним останньому елементу цього проміжку

Інші оператори цієї групи дозволяють порівнювати два впорядкованих списка. Ця група включає наступні оператори:

EQUAL	перша множина є еквівалентною другій. Тобто - найменший елемент (перший) першого операнда дорівнює першому елементу другого операнда, а останні елементи двох операндів також є рівними
CONTAINS	перша множина, представлена першим операндом, включає в себе другу, представлену другим операндом
CONTAINED IN	навпаки - друга множина включає в себе першу
INTERSECTS WITH	перша множина перетинає другу
BEGINS IN	перша множина починається в другій. Це означає, що найменший елемент першої множини належить другій множині
ENDS IN	перша множина закінчується в другій множині. Це означає, що найбільший елемент першої множини належить другій множині
BEGINS BUT NOT ENDS IN	перша множина починається, але не закінчується у другій множині
ENDS BUT NOT BEGINS IN	перша множина закінчується, але не починається у другій множині

Семантика предиката (його інтерпретація) є наступною: перша функція вираховує значення (множину значень) базових складників ротації, які потім подаються в правило в якості першого аргумента. Якщо другий операнд є функцією, тоді проводяться аналогічні підрахунки інших параметрів ротації, що також будуть подані певною величиною або множиною величин у правило в якості другого операнда. Після цього дві величини мають бути порівняні. Якщо дві величини (набори величин) відповідають оператору (2 = 2 або 3 > 2), тоді предикат повертає TRUE (істина), в іншому випадку - результатом буде FALSE (хибно).

В деяких випадках функція може повертати значення NULL. В зв'язку з цим будемо

вважати, що якщо один з операторів має значення NULL, тоді оператор повертає FALSE.

Розглянемо кілька прикладів:

З правила видно те, що гнучкість і сила предиката - це можливість працювати з функціями і операторами. Наявність функцій визначає функціональну повноту і завершеність системи Аудит.

1.2 Заперечення (Predicate negation) і літерали

Якщо ми вводимо в систему аудит певний оператор, то повинні попіклуватися над тим, щоб обов'язково ввести для цього оператора заперечуючий оператор. Це означає, що, вводячи оператор рівності двох операндів ('='), обов'язково для цього оператора необхідно ввести заперечуючий оператор (у нашому випадку це '!='). Вводячи оператор '<=', потрібно не забути ввести оператор "навпаки" - '>', а також наша мова буде включати логічне заперечення NOT, яке необхідно використовувати перед предикатом.

Синтаксис:

<predicate negation>:: NOT <predicate>

Семантика даного виразу наступна: якщо предикат є істинним, то весь вираз повертає значення FALSE і навпаки.

1.3 Кон'юнкція (Conjunctive expressions)

Літерали дозволяють виражати елементарні твердження та заперечення цих тверджень. Але на практиці використовуються набагато складніші твердження, які складаються з декількох елементарних тверджень, з'єднаних логічною зв'язкою. Введемо оператор кон'юнкції AND (логічне &). Кон'юнктивний вираз - це вираз, що складається з літералів, об'єднаних логічною часткою AND. Синтакс такого виразу наступний:

<conjunctive expression>:: <literal-1> AND <literal-2> AND ... AND <literal-n>

При цьому, загальний вираз буде істинним, якщо всі літерали цього виразу також будуть істинними.

1.4 Елементарні правила (Elementary Rule)

1.4.1 Імплікативні елементарні правила (Implicative elementary rules)

Майже всі правил в системі аудит мають наступну форму:

If A then B,

де A та B – це певні твердження, що можуть бути або істинними, або хибними. При цьому всі правила заявляють, що з A випливає B. Істинність результату відповідає істинності логічної формули $A \supset B$.

Ліву частину правила (A) будемо називати передумовою (premise), праву частину (B) – результат (conclusion). Тоді елементарне правило має наступний синтаксис:

<implicative rule>:: If <conjunctive expression-1> Then <conjunctive expression-2>

Семантика: загальне правило буде “істиною”, якщо лише передумова буде хибною або коли і передумова, і результат – “істина”. Іншими словами, імплікативне правило буде хибним, якщо передумова – TRUE, результат – FALSE.

1.4.2 Кон’юнктивні елементарні правила (Conjunctive elementary rules)

Кон’юнктивні елементарні правила відповідають кон’юнктивним виразам, тому вони мають наступний синтаксис:

<conjunctive rule>:: <conjunctive expression>

Необхідно зазначити, що існує така інтерпретація правил, що не залежить від передумови. Для прикладу візьмемо таке формулювання твердження: “у всіх випадках число посадок для одного рейса не повинно перевищувати 8”. Зважаючи на сказане, можна зробити

висновок, що просто необхідно ввести таку конструкцію для побудови анормальних тверджень.

<elementary rule>:: <implicative rule> | <conjunctive rule>

Лише тепер можна зазначити, що елементарні правила по своїй структурі визначені і описані повно.

1.5 Диз'юнктивний набір правил (Disjunctive collection of rules)

Елементарні правила не дозволяють виражати твердження, що мають логічне "АБО" (OR) в передумові чи в результаті. Тому ми вводимо іншу форму, яку назвемо диз'юнктивним набором правил і яка має наступний синтакс:

<disjunctive collection>:: <elementary rule-1> OR ... OR <elementary rule-n>

весь набір правил буде "істиною", коли хоча б одне з елементарних правил буде істинним.

1.6 Загальні правила (General rules)

Загальні правила – це диз'юнктивний набір правил, з'єднаних логічними операторами AND. Синтаксис цього правила:

<general rule>:: <disjunctive collection-1> AND ... AND <disjunctive collection-n>

1.7 Функціональна повнота загальних правил

Одна з головних потреб до будь-якої мови – є її повнота. Є багато визначень повноти мови. Ми розглядаємо повноту, як виразну силу мови. Тут існує два аспекти повноти мови – функціональна і логічна завершеність.

Функціональна завершеність означає, що мова включає в себе всі можливі функції, з допомогою яких можливо виразити будь-яке правило в правовій області. Існує два типи функцій. Перший тип включає в себе функції, що обчислюють певні величини і параметри ротації. Якщо структура ротації фіксована, то зовсім не складно розробити та запрограмувати всі необхідні функції. Другий тип включає в себе функції, що, використовуючи спеціальні алгоритми, обчислюють специфічні параметри, керуючись потенційною мінливістю правил. Логічна завершеність означає повноту мови відносно формальної логіки. Тут треба бути впевненим, що будь-яка формула propositional calculus має еквівалентне відображення в нашій мові.

1.8 Змінні, квантифікатори і закриті (упорядковані) загальні правила

Приймаючи ротацію, як головний вхідний параметр, система розкладає її на складники, з якими потім на пряму може працювати система. Такими складниками, як ми вже говорили, є: чергування, відпочинок, політ, сама ротація і т.д. Оскільки таких елементів по декілька кожного, то для правильної роботи без втрати необхідної інформації, кожному елементу виділяється змінна відповідного типу а потім проходить перевірка на відповідність встановленим правилам по кожній змінній.

Необхідно підкреслити, що система опрацювання ротації на відповідність встановленим правилам завжди працює лише з двома змінними: одна для чергувань, друга для легів. Ці дві величини не є незалежними. Це означає, що змінній для лега може бути назначений лише той лег, який відповідає рейсу, що назначений в даний момент змінній для рейсів.

Виникає питання: як система може працювати лише з двома змінними (для легів та для рейсів), коли сама по собі ротація має набагато більше типів елементів, на які може бути розкладена? Невже такої кількості змінних цілком достатньо? Так! Всі елементи ротації, які на перший погляд являються абсолютно різними і цілком незалежними, можуть бути легко зведені до двох основних типів змінних – лег і рейс. Тобто, якщо два основних елемента знайдені, то інші параметри також легко відшукати.

Отже, ми маємо дві змінні, що використовуються системою аудит для перевірки всієї ротації на відповідність встановленим правилам. Одна – для лега, інша – для рейса, куди входить даний лег. Ми довели, що цих двох змінних цілком достатньо для перевірки всієї ротації. Загальне правило, відпрацювавши певний елемент, повертає одне з двох можливих значень – TRUE (правда) або FALSE (хибно). Така відповідь означає, що ротація задовольняє або не задовольняє відповідно встановленим законодавством нормам польотів.

Проте, яке ж рішення повинна винести система, обробивши всі змінні? Яким чином результат обробки кожної пари елементів ротації зв'язується з результуючим варіантом?

Для відповіді на наступне питання ми використовуємо квантифіктори змінної. Універсальний квантифікатор ALL означає, що перевірка на відповідність законодавчим потребам повинна завершитись успішно для всіх значень змінної. Квантифікатоор EXIST означає, що відповідність правилам має бути хоча б для одного значення змінної.

Для того, щоб приєднати створені необхідністю квантифікатори до нашої мови правил польоту, існує така синтаксична конструкція:

<closed ordered set of general rules>:: <duty quantifier> <leg quantifier> <ordered set of general rules>

Очевидно, що при відсутності у загальних правилах предикатів, що описують леги (це означає, що загальні правила не містять функцій, які обчислюють параметри легів), тоді квантифікатотр лега також відсутній. Якщо ж загальне правило в своїй структурі не містить предикатів, що описують і леги, і рейси (тобто загальне правило містить лише функції, які обраховують параметри ротації в цілому), тоді обидва квантифікатора відсутні. Проте, якщо квантифікатор лега необхідний, то обов'язково необхідно використовувати також квантифікатотр чергування.

1.9 Повнота закритих загальних правил у відношенні до предикатного числення

Тепер обсудимо повноту закритих загальних правил у відношенні до предикатного числення. Ми не повинні розглядати впорядковану систему загальних правил, тому що, як ми бачили у попередній секції, така множина повинна бути представлена у формі невпорядкованої множини правил. З точки зору предикатного числення, існує три основні обмеження нашої мови.

По-перше, ми домовились, що існує лише одна змінна для лега і одна для рейса всередині загального правила (у предикатному ж численні ми можемо використовувати будь-яку кількість змінних для всіх типів елементів ротації).

По-друге, наша змінна для лега не є незалежною. Вона може приймати лише значення тих легів, які містяться в рейсі, назначеному змінній для рейсів.

І по-третє, був жорстко встановлений порядок задання квантифікаторів – спочатку квантифікатор для чергування, потім квантифікатор для лега і ніяк по-іншому.

1.10 Систем правил (Rule system)

В загальному випадку ротація перевіряється по черзі усіма загальними правилами в наборі. Тому було б корисним ввести в нашу мову поняття системи правил, яке буде означати множину закритих впорядкованих основних правил, з'єднаних логічним оператором AND. Тобто:

<rule system>:: <closed ordered set general rules-1> AND ... AND <closed ordered set of general rules-n>