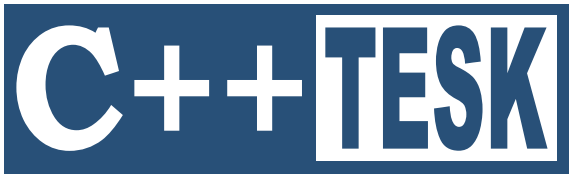

C++TESK Testing ToolKit: **Обходчики**

Версия 1.0, 23/05/2011

Набор инструментов C++TESK Testing ToolKit можно скачать на странице http://forge.ispras.ru/projects/cpptesk-toolkit.

Вопросы по использованию C++TESK Testing ToolKit, а также описание обнаруженных проблем в инструментах и документации к ним отправляйте по адресу cpptesk-support@ispras.ru. Для этого также можно использовать форум http://hw-forum.ispras.ru.

Содержание

Введение

В документе описываются *обходчики* (*test engines*), входящие в состав набора инструментов C++TESK Testing ToolKit (далее C++TESK). *Обходчик* является ядром *генератора стимулов*, компонента *тестовой системы*, отвечающего за построение последовательности стимулов (*тестовых воздействий*) на *целевую систему*. Стимулы представляются собой вызовы операций целевой системы с определенными значениями параметров; последовательность стимулов называется *тестовой последовательностью*. Входной информацией для обходчика является *тестовый сценарий* — высокоуровневая спецификация теста, определяющая доступные для тестирования стимулы. Тестовая последовательность строится путем интерпретации обходчиком заданного сценария. Важно отметить, что обходчики имеют одинаковый интерфейс, что позволяет использовать разные обходчики для выполнения одного и того же тестового сценария.

Библиотека C++TESK включает два обходчика: `fsm` (*Finite State Machine*) и `rnd` (*Random*), расположенные в пространстве имен `cpptesk::ts::engine`. Первый из них, `fsm`, осуществляет обход *графа состояний* конечного автомата, описанного в неявной форме в тестовом сценарии. Критерием завершения тестирования в этом случае является посещение всех состояний графа достижимых из начального состояния (что подразумевает проход по всем дугам, выходящим из достижимых состояний). Обходчик `rnd` строит тестовую последовательность *случайным* образом — на каждом шаге выбирается случайный стимул или набор стимулов из числа описанных в тестовом сценарии. Тестирование прекращается при выполнении заданного числа шагов. Подробнее указанные обходчики описываются в соответствующих разделах документа.

Перед прочтением дальнейшей части документа рекомендуется познакомиться с созданием тестовых сценариев с помощью C++TESK. Эта информация содержится, например, в документе «*C++TESK Hardware Edition: Краткий справочник*» (раздел «*Создание тестового сценария*»).

Структура тестового сценария

С точки зрения обходчика, тестовый сценарий состоит из двух основных частей: (1) *итератора стимулов* и (2) *функции вычисления состояния*.

Итератор стимулов

Итератор стимулов задается *сценарными методами* сценарного класса. Каждый сценарный метод итерирует параметры некоторого стимула, используя обычные конструкции построения циклов, имеющиеся в C++ (`for`, `while` и т.п.). Итерации осуществляются по *итерационным переменным* — полям *контекста итерации*, передаваемого в качестве параметра сценарного метода. Для доступа к итерационным переменным используется макрос `CPPTESK_ITERATION_VARIABLE(`*`имя`*`)`. Итерации начинаются с макроса `CPPTESK_ITERATION_BEGIN` и заканчиваются макросом `CPPTESK_ITERATION_END`. Внутри системы циклов располагается блок `CPPTESK_ITERATION_ACTION{...}`, реализующий *подачу стимула* на целевую систему. Подача стимула заканчивается вызовом макроса `CPPTESK_ITERATION_YIELD(`*`вердикт`*`)`.

Типичный сценарный метод имеет следующую структуру:

```
bool MyScenario::ScenarioMethod(AbcCtx &ctx) {
    int &a = CPPTESK_ITERATION_VARIABLE(a);
    ...
    int &z = CPPTESK_ITERATION_VARIABLE(z);
```

```
        CPPTESK_ITERATION_BEGIN
        for(a = 0; a < Na; a++)
        ...
        for(z = 0; z < Nz; z++) {
            CPPTESK_ITERATION_ACTION {
                ...
                CPPTESK_ITERATION_YIELD(...);
            }
        }
        CPPTESK_ITERATION_END
    }
```

Сценарный метод *ScenarioMethod* для заданного состояния эталонной модели (состояния спецификации) *ModelState* порождает набор стимулов, идентифицируемых целыми числами из отрезка $[0, N_{max}(ScenarioMethod, ModelState)-1]$, где $N_{max}(ScenarioMethod, ModelState)$ — число выполнений блока `CPPTESK_ITERATION_ACTION` сценарного метода для заданного состояния эталонной модели.

Совокупный итератор стимулов тестового сценария получается «объединением» итераторов, определяемых в сценарных методах. Итератор стимулов тестового сценария описывает множество всех возможных стимулов и вводит глобальную для данного тестового сценария систему идентификации стимулов: для фиксированного состояния эталонной модели *ModelState*, стимулы идентифицируются целыми числами из отрезка $[0, N_{max}(ModelState)-1]$, где $N_{max}(ModelState)=\sum_{i=0,n-1}\{N_{max}(ScenarioMethod_i, ModelState)\}$. При нумерации стимулов учитывается порядок регистрации сценарных методов.

Функция вычисления состояния

Функция вычисления состояния определяется в *методе вычисления состояния* сценарного класса. Тип возвращаемого значения этого метода может быть скалярным (`int`, `float` и т.п.) или строковым (`std::string`). Возвращаемое значение интерпретируется как состояние эталонной модели на некотором уровне абстракции.

Функция вычисления состояния является необязательным компонентом тестового сценария, который используется только обходчиком `fsm`.

Обходчики

Работа обходчиков управляется параметрами командной строки, некоторые из которых являются общими для всех обходчиков:

- `--cpu` — идентификатор вычислительного узла, на котором запускается тест[1];
- `--length` — длина тестовой последовательности[2].

Общие параметры командной строки

Параметр cpu

Параметр `--cpu` *`целое_число`* задает идентификатор вычислительного узла (компьютера, микропроцессора или ядра), на котором запускается тест. Параметр используется в случае распределенного запуска теста на нескольких вычислительных узлах. Основное назначение параметра заключается в создании различных вариаций выполнения теста на разных узлах

[1] Этот параметр имеет смысл при распределенном запуске теста.

[2] В существующей версии C++TESK Testing ToolKit обходчик `fsm` игнорирует параметр `--length`.

вычислительной системы. В частности, он используется для инициализации генератора случайных чисел, а также при выборе очередного стимула (дуги графа) обходчиком `fsm`.

Параметр `length`

Параметр `--length` *`целое_число`* ограничивает длину генерируемой обходчиком тестовой последовательности. Как только длина становится равной заданному числу, тестирование прекращается.

Значения по умолчанию

По умолчанию параметры `--cpu` и `--length` принимают следующие значения:

```
--cpu 0 --length 1000
```

Обходчик `fsm`

Обходчик `fsm` генерирует тестовую последовательность на основе обхода графа состояний конечного автомата, описанного в неявной форме в тестовом сценарии. Критерием завершения тестирования является посещение всех состояний графа достижимых из начального состояния и проход по всем дугам, выходящим из них.

Краткое описание алгоритма работы

Обходчик `fsm` работает следующим образом. На каждом шаге тестирования вычисляется текущее состояние. При этом возможны два варианта: (1) в текущем состоянии есть непройденные дуги[3] и (2) все дуги, выходящие из текущего состояния, пройдены. В первом случае обходчик выбирает одну дугу из числа непройденных, подает на целевую систему соответствующий стимул, после чего тестирование продолжается. Во втором случае возможны две альтернативы: (2.1) существуют другие состояния (из числа пройденных ранее), в которых есть непройденные дуги, и (2.2) все известные дуги пройдены. В первом случае обходчик ищет путь в состояние, в котором есть непройденные дуги. После этого ситуация (2.1) сводится к ситуации (1). Во втором случае тестирование прекращается.

Параметры командной строки

Обходчик `fsm` поддерживает следующие опции командной строки:

- `--randomize` — рандомизация выбора дуги;
- `--nondeterministic` — поддержка недетерминированных графов;
- `--dump-fsm` — сохранение графа состояний в файле после окончания тестирования:
  - `--full-graph` — сохранение всего графа;
  - `--forward-tree` — сохранение остовного дерева.

Опция `randomize`

Опция `--randomize` включает рандомизацию выбора очередной дуги. Это означает, что каждый раз, когда у обходчика есть выбор, какую дугу использовать, выбор осуществляется случайным образом. Если данная опция не установлена, дуги выбираются в том порядке, в котором они описаны в тестовом сценарии.

---

[3] Правильнее говорить не о *дугах*, а о *стимулах*, поскольку в случае *недетерминизма* одному стимулу может соответствовать несколько дуг, выходящих из одного состояния.

Опция `nondeterministic`

Опция `--nondeterministic` включает поддержку *недетерминированных* графов, то есть графов, в которых из одного состояния может исходить несколько дуг, помеченных одним стимулом. Основная сложность работы с недетерминированными графами заключается в том, что тестовая система не может достоверно определить, какая дуга будет пройдена при подаче того или иного стимула, что усложняет поиск пути в состояния, в которых есть непройденные дуги.

Установка опции `--nondeterministic` разрешает использовать недетерминированные дуги при поиске путей (расширяя тем самым класс поддерживаемых графов). Если данная опция не установлена, пути строятся только по детерминированным дугам.

Опция `dump-fsm`

Опция `--dump-fsm` используется для сохранения графа состояний в файле после окончания тестирования. Файл имеет фиксированное имя — `fsm.gv`; данные в нем представлены в формате Graphviz — открытого пакета визуализации графов (http://www.graphviz.org).

Опция `full-graph`

Опция `--full-graph` (установленная совместно с `--dump-fsm`) сохраняет граф состояний целиком.

Замечание: опции `--full-graph` и `--forward-tree` несовместимы.

Опция `forward-tree`

Опция `--forward-tree` (установленная совместно с `--dump-fsm`) сохраняет только остовное дерево графа состояний.

Замечание: опции `--forward-tree` и `--full-graph` несовместимы.

Значения по умолчанию

По умолчанию опции `–-randomize`, `--nondeterministic` и `--dump-fsm` не установлены.

Если задана опция `--dump-fsm`, по умолчанию граф состояний сохраняется целиком.

Обходчик `rnd`

Обходчик `rnd` строит тестовую последовательность *случайным* образом — на каждом шаге выбирается случайный стимул или набор стимулов из числа описанных в тестовом сценарии. Тестирование прекращается при достижении заданной длины тестовой последовательности.

Краткое описание алгоритма работы

Существует два режима работы обходчика `rnd`: *последовательный* (см. раздел «*Опция `sequential`*») и *параллельный* (см. раздел «*Опция `parallel`*»).

В последовательном режиме все стимулы, задаваемые итератором тестовых воздействий, равноправны. На каждом шаге тестирования идентификатор стимула выбирается случайным образом из отрезка [0, N_{max}(*ModelState*)-1], после чего соответствующий стимул применяется.

В параллельном режиме стимул с идентификатором 0 имеет особый смысл — это временн*ая* задержка; во всех остальных стимулах время не изменяется (это «мгновенные» воздействия без ожидания реакций на них). В таком режиме на каждом шаге тестирования составляется случайная последовательность стимулов (*мультистимул*), заканчивающаяся стимулом с

идентификатором 0. Другими словами, параллельно подается несколько стимулов, после чего делается задержка.

Размер составляемого мультистимула зависит от *режима загрузки*. Обходчик `rnd` поддерживает четыре режима загрузки:

- *режим минимальной загрузки* (см. раздел «*Опция* `min-load`»);
- *режим максимальной загрузки* (см. раздел «*Опция* `max-load`»);
- *режим фиксированной загрузки* (см. раздел «*Параметр* `fix-load`»);
- *режим переменной загрузки* (см. раздел «*Опция* `var-load`»).

Критерием завершения тестирования является выполнение заданного числа шагов тестирования (см. раздел «*Параметр* `length`»).

Параметры командной строки

Обходчик `rnd` поддерживает следующие параметры командной строки:

- `--sequential` — последовательный режим;
- `--parallel` — параллельный режим:
  - `--min-load` — режим минимальной загрузки;
  - `--max-load` — режим максимальной загрузки;
  - `--fix-load` — режим фиксированной загрузки;
  - `--var-load` — режим переменной загрузки.

Опция `sequential`

Опция `--sequential` включает *последовательный режим* работы обходчика. В этом режиме стимул на каждом шаге тестирования выбирается случайным образом из числа описанных в тестовом сценарии. Последовательный режим предназначен в первую очередь для систем, в которых невозможна параллельная подача нескольких стимулов.

Замечание: опции `--sequential` и `--parallel` несовместимы.

Опция `parallel`

Опция `--parallel` включает *параллельный режим* работы обходчика. В этом режиме на каждом шаге тестирования подается случайная последовательность стимулов, заканчивающаяся стимулом с идентификатором 0 — временной задержкой[4]. Предполагается, что все стимулы, за исключением стимула 0, выполняются мгновенно — они запрашивают запуск той или иной операции, но реальный запуск осуществляется только при вызове стимула 0, когда происходит сдвиг времени. Шаг тестирования в параллельном режиме схематично показан на Рис. 1.

[4] Обычно этот стимул оформляется в виде сценарного метода без итераций, называемого `nop()` или `delay()`.

$$\ldots \rightarrow \left\{ \begin{array}{c} стимул\ i_1(t) \\ стимул\ i_2(t) \\ \ldots \\ стимул\ i_{n(t)}(t) \end{array} \right\} \rightarrow стимул\, 0\,(задержка) \rightarrow \ldots,\ i_k(t) \neq 0.$$

Рисунок 1. Шаг тестирования в параллельном режиме

Замечание: опции `--parallel` и `--sequential` несовместимы.

Опция `min-load`

Опция `--min-load` включает режим *минимальной* загрузки — на каждом шаге тестирования подается не более одного стимула. Данный режим обычно используется для отладки тестовой системы.

Опция `max-load`

Опция `--max-load` включает режим *максимальной* загрузки — на каждом шаге тестирования подаются все возможные стимулы, описанные в тестовом сценарии.

Параметр `fix-load`

Параметр `--fix-load` *`процент_загрузки`* задает режим *фиксированной* загрузки — на каждом шаге тестирования подается некоторая фиксированная доля стимулов. Например, если указать `--fix-load 50`, на каждом шаге будет параллельно подаваться половина стимулов.

Опция `var-load`

Опция `--var-load` включает режим *переменной* загрузки — при тестировании загрузка целевой системы плавно возрастает от минимальной до максимальной.

Значения по умолчанию

По умолчанию установлена опция `--sequential`.

Если задана опция `--parallel`, по умолчанию используется режим переменной загрузки `--var-load`.