Модел. и анализ информ. систем. Т. **21**, № **4** (2014) 75–90

УДК 517.9

Моделирование согласованного поведения ПЛК-датчиков

Кузьмин Е. В.[1], Рябухин Д. А., Соколов В. А.

Ярославский государственный университет им. П. Г. Демидова
150000 Россия, г. Ярославль, ул. Советская, 14

e-mail: {*kuzmin,sokolov*}*@uniyar.ac.ru, dmitriy_ryabukhin@mail.ru*

получена 4 августа 2014

Ключевые слова: программирование логических контроллеров, моделирование, спецификация и верификация программ, моделирование датчиков

Статья продолжает цикл работ, посвященный разработке подхода к построению и верификации «дискретных» программ логических контроллеров (ПЛК), обеспечивающего возможность анализа их корректности с помощью метода проверки модели (model checking). Подход получил название «Программирование и верификация по LTL-спецификации».

При верификации ПЛК-программы методом проверки модели возникает необходимость в построении ее конечной модели. При этом для успешной проверки соответствия модели требуемым программным свойствам важно учитывать то, что далеко не все комбинации входных сигналов от датчиков могут встречаться в действительности при работе ПЛК с объектом управления. Этот факт требует более внимательного отношения к построению модели программы ПЛК.

В статье предлагается описывать согласованное поведение датчиков с помощью трех групп LTL-формул, которые при проверке справедливости программных свойств будут оказывать влияние на программную модель, приближая ее к реальному поведению исходной ПЛК-программы. Идея LTL-требований демонстрируется на примере.

Программа ПЛК представляет собой описание реакций на входные сигналы от датчиков, переключателей и кнопок. Предложенный подход к моделированию согласованного поведения ПЛК-датчиков при построении модели программы ПЛК позволяет сосредоточиться на моделировании именно этих реакций по тексту программы без внедрения в код модели дополнительных конструкций, призванных отразить реалистичное поведение датчиков. Согласованное поведение датчиков учитывается лишь на стадии проверки соответствия программной модели требуемым свойствам, т. е. доказательство выполнимости свойств для построенной модели происходит с условием, что рассматриваемая модель содержит только те исполнения исходной программы, которые отвечают требованиям согласованного поведения датчиков.

[1]Работа проводилась при финансовой поддержке РФФИ, грант №12-01-00281-а.

Введение

Применение программируемых логических контроллеров (ПЛК) в системах управления сложными производственными процессами предъявляет строгие требования корректности к программам ПЛК. Любая программная ошибка считается недопустимой. Вместе с тем программирование ПЛК представляет собой прикладную область, в которой существующие наработки в области формальных методов моделирования и анализа программных систем могут иметь успешное применение.

Эта статья продолжает цикл работ [1, 2, 3, 4, 5], посвященный разработке подхода к построению и верификации «дискретных» ПЛК-программ, обеспечивающего возможность анализа их корректности с помощью метода проверки модели (model checking) [7]. Подход получил название «Программирование и верификация по LTL-спецификации». При этом подходе в качестве языка спецификации программного поведения используется язык темпоральной логики LTL. Анализ корректности LTL-спецификации относительно программных свойств производится с помощью программного средства символьной проверки модели Cadence SMV [10].

ПЛК — «реагирующая» система, имеющая множество входов, подключаемых посредством датчиков к объекту управления, и множество выходов, подключаемых к исполнительным устройствам [9, 6]. Программа ПЛК выполняется в рабочем цикле: считывание входов, выполнение программы и выставление выходов.

При верификации ПЛК-программы методом проверки модели возникает необходимость в построении ее конечной модели. При этом для успешной проверки соответствия модели требуемым программным свойствам важно учитывать то, что далеко не все комбинации входных данных (сигналов от датчиков) могут встречаться в действительности при работе ПЛК с объектом управления. Этот факт требует более внимательного отношения к построению модели программы ПЛК.

В этой статье предлагается описывать согласованное поведение датчиков с помощью трех групп LTL-формул, которые при проверке справедливости программных свойств будут оказывать влияние на программную модель, приближая ее к реальному поведению исходной ПЛК-программы. Для каждого датчика первая группа LTL-формул отвечает на вопрос, какими должны быть значения других датчиков на входах ПЛК, если рассматриваемый датчик «горит» и если он «молчит». Вторая группа LTL-требований к поведению датчика касается случаев срабатывания датчика и случаев снятия им единичного сигнала. С помощью LTL-формул дается объяснение того, что может являться причиной срабатывания датчика и при каких условиях это вообще возможно. Третья группа LTL-формул объясняет возможные причины, а также описывает цепь событий, которые приводят к тому, что датчик навсегда остается только в одном из своих состояний, т. е. когда датчик навсегда остается «гореть» или же он навсегда «замолкает». Третья группа LTL-требований позволяет исключить из программной модели нереалистичное «залипание» датчиков, т. е. не рассматривать «нечестные программные исполнения», в которых датчик более никогда не меняет своего состояния при том, что ситуации, позволяющие сделать это, появляются бесконечно часто.

Идея трех групп LTL-требований (по два требования в каждой группе) для описания согласованного поведения датчиков демонстрируется на примере задачи «Установка для приготовления смесей».

1. Метод проверки модели для ПЛК-программ

Задача проверки модели (Model Checking) состоит в определении выполнимости для конечной модели программной системы (в виде структуры Крипке) свойства, выраженного формулой темпоральной логики.

Структурой Крипке над множеством элементарных высказываний P называется система переходов $\mathcal{S} = (S, s_0, \to, L)$, где S — конечное множество состояний (модели программной системы), $s_0 \in S$ — начальное состояние, $\to \subseteq S \times S$ — отношение переходов, $L : S \to 2^P$ — функция, помечающая каждое состояние множеством элементарных высказываний, истинных в этом состоянии.

Путь в структуре Крипке из состояния s_0 — это бесконечная последовательность состояний $\pi = s_0 s_1 s_2 \ldots$ такая, что $\forall\, i \geq 0$ выполняется $s_i \to s_{i+1}$.

В качестве языка спецификации поведенческих свойств программной модели рассматривается язык темпоральной логики линейного времени LTL (Linear-Time Temporal Logic). Выбор логики LTL связывается с тем, что программа ПЛК является классической реактивной (реагирующей) управляющей системой, которая, будучи однажды запущенной, должна иметь корректное бесконечное поведение. Условия корректности удобно задавать в виде шаблонов свойств, которым должны соответствовать корректные исполнения программы. В темпоральной логике LTL каждая формула по сути представляет собой такой шаблон.

Формулы логики LTL строятся по следующей грамматике при $p_i \in P$:

$$\varphi, \psi ::= \mathit{true} \mid p_0 \mid p_1 \mid \ \ldots \ \mid p_n \mid \neg\varphi \mid \psi \wedge \varphi \mid X\varphi \mid \psi\, U\, \varphi \mid F\,\varphi \mid G\,\varphi.$$

Формула логики LTL описывает свойство одного пути структуры Крипке, выходящего из некоторого выделенного текущего состояния. Темпоральные операторы X, F, G и U имеют следующую интерпретацию: $X\varphi$ означает, что формула φ должна выполняться в следующем состоянии, $F\varphi$ — φ должна выполняться в некотором будущем состоянии пути, $G\varphi$ — φ должна выполняться в текущем состоянии и во всех будущих состояниях пути, $\psi\, U\, \varphi$ — φ должна выполняться в текущем или будущем состоянии при том, что во всех состояниях (начиная с текущего) до этого момента должна выполняться формула ψ. Операторы F и G являются производными и вводятся для удобства спецификации свойств: $F\,\varphi = \mathit{true}\, U \varphi$, $G\,\varphi = \neg F \neg\varphi$. Кроме того, далее будут использоваться классические логические связки $\vee$ и $\Rightarrow$.

Структура Крипке удовлетворяет формуле (свойству) φ логики LTL, если φ выполняется для всех путей, выходящих из начального состояния s_0.

Модель Крипке для программы ПЛК строится естественным образом. Состояние модели — это состояние программы (вектор значений всех переменных) ПЛК после одного полного прохода рабочего цикла. Начальное состояние модели — состояние программы после инициализации значений переменных. Таким образом, в качестве перехода из одного состояния модели в другое (или то же самое) рассматривается полное исполнение программы за один проход рабочего цикла ПЛК.

Отметим, что модель от исходной ПЛК-программы отличается лишь дискретным представлением работы таймеров (см. [5]). Если таймеры в программе не применяются, то поведение модели полностью совпадает с поведением программы.

В качестве элементарных высказываний модели будут рассматриваться логические (с применением арифметических операторов и операторов сравнения) выражения над переменными ПЛК-программы.

2. Согласованное поведение датчиков

Рассмотренная в предыдущем разделе модель программы ПЛК, как правило, нуждается в дополнительной более тонкой настройке, чтобы она могла максимально близко соответствовать реальному поведению ПЛК-программы. Например, при работе с таймерами из программной модели (структуры Крипке) необходимо исключить те пути (программные исполнения), в которых включенный таймер никогда не срабатывает (см. [5]). Более того, часто требуется исключить нереалистичные комбинации значений входных переменных, через которые отслеживаются состояния датчиков. Например, при исправной работе датчиков их реакция (комбинация значений), при которой в баке с жидкостью датчик верхнего уровня «горит», а датчик нижнего уровня «молчит», считается нереалистичной.

Для описания реалистичных путей программной модели, учитывающих согласованное поведение датчиков, удобно использовать темпоральную логику LTL. В данной работе предлагается описывать требования к поведению каждого датчика шестью LTL-формулами – три группы по две формулы.

Первая группа LTL-требований к поведению датчика отвечает на вопрос, какими должны быть значения других датчиков на текущем рабочем цикле ПЛК, соответственно 1) если рассматриваемый датчик «горит» и 2) если он «молчит». Шаблоны LTL-требований для переменной-датчика S представлены ниже:

$$\mathbf{G}(\ \mathbf{X}(\ \ S) \Rightarrow \mathit{Cond1}\ ), \tag{1}$$

$$\mathbf{G}(\ \mathbf{X}(\neg S) \Rightarrow \mathit{Cond1}'\ ), \tag{1'}$$

где условия *Cond1* и *Cond1′* описывают соответствующие ограничения на допустимые значения других датчиков. Датчик S находится под действием темпорального оператора $\mathbf{X}$ для того, чтобы в условиях *Cond1* и *Cond1′* можно было обращаться не только к текущим значениям программных переменных, но и к значениям, которые переменные имели на предыдущем рабочем цикле, т. е. на предыдущем шаге.

Вторая группа LTL-требований к поведению датчика касается случаев срабатывания датчика и случаев снятия им единичного сигнала. LTL-формулы требований объясняют, что может являться причиной этого и при каких условиях вообще возможно совершение соответствующего действия (срабатывания):

$$\mathbf{G}(\ \neg S \wedge \mathbf{X}(\ \ S) \Rightarrow \mathit{Cond2}\ ), \tag{2}$$

$$\mathbf{G}(\quad S \wedge \mathbf{X}(\neg S) \Rightarrow \mathit{Cond2}'\ ). \tag{2'}$$

Третья группа LTL-требований к поведению датчика объясняет возможные причины, а также описывает цепь событий, которые приводят к тому, что датчик навсегда остается только в одном из своих состояний, т. е. либо датчик будет всегда «гореть», либо он навсегда «замолкает»:

$$\mathbf{G}(\ \mathbf{G}(\ \ S) \Rightarrow \mathit{Cond3}\ ), \tag{3}$$

$$\mathbf{G}(\ \mathbf{G}(\neg S) \Rightarrow \mathit{Cond3}'\ ). \tag{3'}$$

Третья группа LTL-требований позволяет исключить из программной модели пути, связанные с нереалистичным «залипанием» датчиков, т. е. не рассматривать «нечестные» пути, в которых датчик более никогда не меняет своего состояния при том, что ситуации, позволяющие сделать это, появляются бесконечно часто.

Продемонстрируем идею этих трех групп LTL-требований на примере описания

поведения датчика первого уровня резервуара LS1 из задачи «Установка для приготовления смесей», которая будет подробно рассмотрена в следующем разделе. Далее LTL-формулы будут записываться на языке верификатора Cadence SMV. На языке SMV символы «&», «|», «∼» и «->» означают соответственно логические «и», «или», «не» и импликацию.

Первая группа LTL-требований для датчика LS1 имеет следующий вид:

```
G(  X(LS1) ->  X(LS0) );
G( ~X(LS1) -> ~X(LS2) ).
```

Эти формулы означают, что если датчик первого уровня резервуара «горит», то должен «гореть» и датчик нулевого уровня резервуара LS0, а если датчик первого уровня «молчит», то должен «молчать» и датчик второго уровня LS2.

Вторая группа LTL-требований для датчика LS1 описывает случаи изменения его состояния (выставление и снятие единичного сигнала):

```
G(~LS1 &  X(LS1) -> LS0 & X(LS0) & ~LS2 & ~X(LS2) & (Vlv1 & TS1 | Vlv2 & TS2));
G( LS1 & ~X(LS1) -> LS0 & X(LS0) & ~LS2 & ~X(LS2) & (EVlv       | PVlv      )).
```

Первая формула означает, что если датчик первого уровня резервуара LS1 сработал, то либо был открыт первый клапан Vlv1 и первый бак не был пустым («горит» датчик уровня первого бака TS1), либо был открыт второй клапан Vlv2 и второй бак не был пустым («горит» датчик уровня второго бака TS2), а также датчик нулевого уровня резервуара LS0 «горел» и продолжает «гореть» и датчик второго уровня резервуара LS2 «молчал» и продолжает «молчать», т. е. в резервуар была осуществлена подача смешиваемого компонента с учетом того, что за время одного рабочего цикла ПЛК объем содержимого резервуара не может измениться более чем на один уровень. Вторая формула говорит о том, что если датчик LS1 погас, значит, либо аварийный клапан EVlv, либо разливной клапан PVlv находился в открытом состоянии, датчик LS2 «молчал» и продолжает «молчать», датчик LS0 «горел» и продолжает «гореть», т. е. осуществлялся слив содержимого резервуара с учетом того, что за время одного рабочего цикла его объем не может измениться более чем на один уровень.

Третья группа LTL-требований касается «залипания» датчика LS1 и имеет вид:

```
G( G( LS1) -> F(G~(EVlv | PVlv)) |
              G((EVlv | PVlv) -> F(Vlv1 & TS1 | Vlv2 & TS2)) );
G( G(~LS1) -> F(G~(Vlv1 & TS1 | Vlv2 & TS2)) |
              G((Vlv1 & TS1 | Vlv2 & TS2) -> F(EVlv | PVlv)) ).
```

Если датчик LS1 навсегда остается в «горящем» состоянии, значит, либо 1) через некоторое время ни аварийный клапан EVlv, ни разливной клапан PVlv никогда более не будут открыты, либо 2) всякий раз, когда аварийный клапан или разливной клапан находится в открытом состоянии, рано или поздно в открытом состоянии будет находиться первый клапан Vlv1 при непустом первом баке или второй клапан Vlv2 при непустом втором баке, т. е. более не выполняются условия снятия сигнала или же всякий раз, когда осуществляется слив содержимого резервуара, происходит также и подача компонента в резервуар. Если датчик LS1 «замолкает» («гаснет») навсегда, значит, либо 1) через некоторое время ни первый клапан при непустом первом баке, ни второй клапан при непустом втором баке никогда более не будут открыты, либо 2) всякий раз, когда в открытом состоянии находится первый клапан

при непустом первом баке или второй клапан при непустом втором баке, рано или поздно в открытом состоянии также будет находиться аварийный клапан или разливной клапан, т. е. более не выполняются условия срабатывания датчика LS1 или же всякий раз, когда осуществляется подача компонента в резервуар, происходит также и слив содержимого резервуара.

3. Установка для приготовления смесей

Установка, схема которой представлена на рис. 1, предназначена для приготовления смесей из двух компонентов. Первый смешиваемый компонент подается из бака 1 посредством клапана «Кл1» в изначально пустой (нет сигнала от датчика «ДУ0») резервуар до тех пор, пока не сработает датчик уровня «ДУ1». По срабатыванию датчика «ДУ1» клапан «Кл1» закрывается. Затем открывается «Кл2» и из бака 2 подается второй смешиваемый компонент до тех пор, пока не сработает датчик уровня «ДУ2», после чего клапан «Кл2» закрывается. Далее в течение T секунд с помощью мешалки в резервуаре происходит перемешивание компонентов. Функционирование мешалки определяется по датчику работы привода мешалки «ДРП». По истечении этого времени открывается клапан «Кл0» и происходит выгрузка готовой смеси. Выгрузка заканчивается, т. е. клапан «Кл0» закрывается, как только пропадает сигнал от датчика уровня «ДУ0». Клапан «Кл3» используется для аварийного слива некондиционной смеси. Наличие смешиваемых компонентов в баках определяется по соответствующим датчикам уровня «ДБ1» и «ДБ2».

На панели управления элементы «Ав. клап.», «Кл. 2», «Кл. 1», «Кл. 0» и «Привод» являются кнопками с самофиксацией, т. е. переключателями. С помощью этих переключателей выставляются входные сигналы для ПЛК на открытие/закрытие клапанов и запуск/остановку привода мешалки.

Установка по приготовлению смесей является автоматической. Управление установкой осуществляется с помощью ПЛК, получающего входные сигналы от датчиков установки и переключателей панели управления оператора и подающего выходные сигналы на приводы установки и лампы панели управления (см. рис. 1).

Задача состоит в написании программы для ПЛК с 11 входами и 11 выходами, предназначенного для управления установкой. Интерфейс ПЛК управления установкой представлен на рис. 2. Предполагается, что сигналы от датчиков на соответствующие лампы панели управления идут напрямую, минуя ПЛК.

Требования к программе ПЛК:

1. Предполагается, что неисправная работа датчиков и клапанов исключена. При поступлении выходного сигнала на клапан он считается открытым, если же этот сигнал снимается, клапан считается закрытым. Датчики уровня отражают реальный объем компонентов в баках и смеси в резервуаре. Датчик работы привода мешалки определяет действительное состояние мешалки. В отличие от клапанов установленный сигнал на запуск привода мешалки «Мешалка включена» еще не означает, что мешалка работает.

2. Лампа «Компонент 1» должна загораться (выставляется выходной сигнал контроллера), если резервуар для смешивания не пуст и была подача первого смешиваемого компонента из непустого бака 1. Лампа должна гаснуть, как только

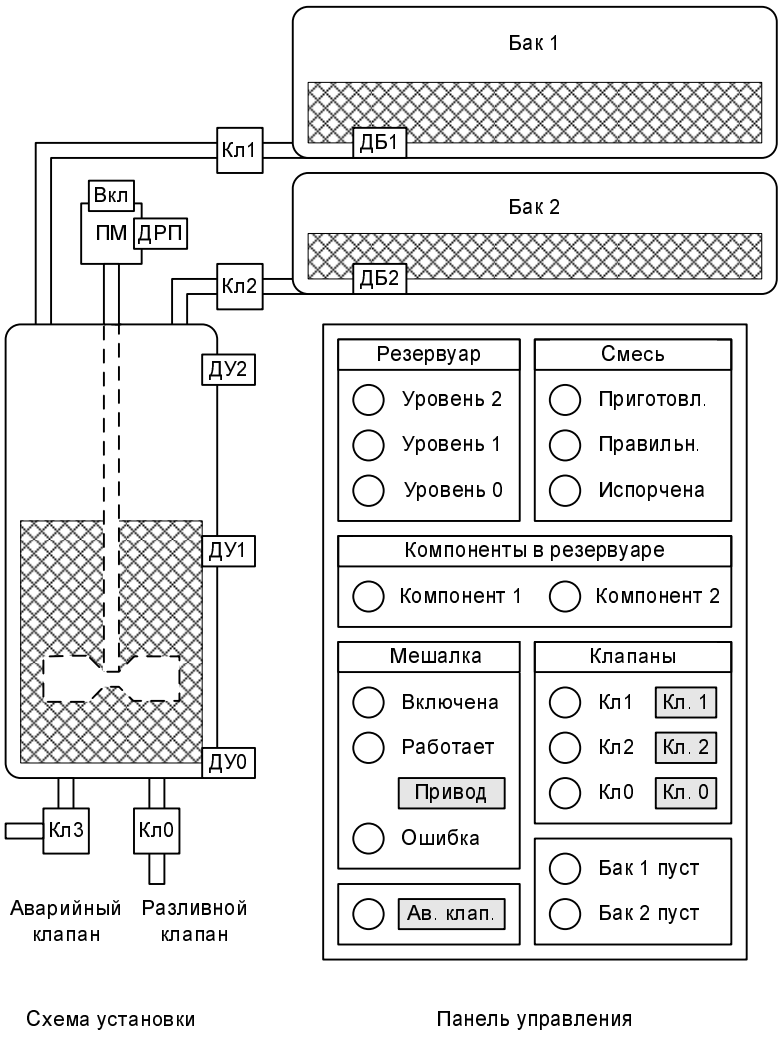

Рис. 1. Схема и панель управления установки для приготовления смесей

резервуар пустеет. Аналогичным образом выставляется и снимается сигнал ПЛК на лампу «Компонент 2».

3. Смесь в резервуаре считается испорченной (выставляется сигнал на лампу «Смесь испорчена»), если либо 1) в правильную смесь (горит лампа «Смесь правильн.») были добавлены компоненты из баков, либо 2) произошел слив части неоднородной неправильной смеси, либо 3) в смесь, объем которой в резервуаре превышает уровень 1, был добавлен первый компонент, либо 4) в смесь, не достигшую в резервуаре уровня 1 или же прошедшую отметку уровня 2, был добавлен второй компонент. Испорченная смесь перестает считаться испорченной, если она полностью отсутствует в резервуаре или состоит только из первого компонента, находясь при этом ниже датчика уровня 1.

4. Смесь в резервуаре считается правильной (выставляется сигнал на лампу «Смесь правильн.»), если в смеси соблюдены пропорции первого и второго смешиваемых компонентов. Правильная смесь перестает быть правильной (снимается

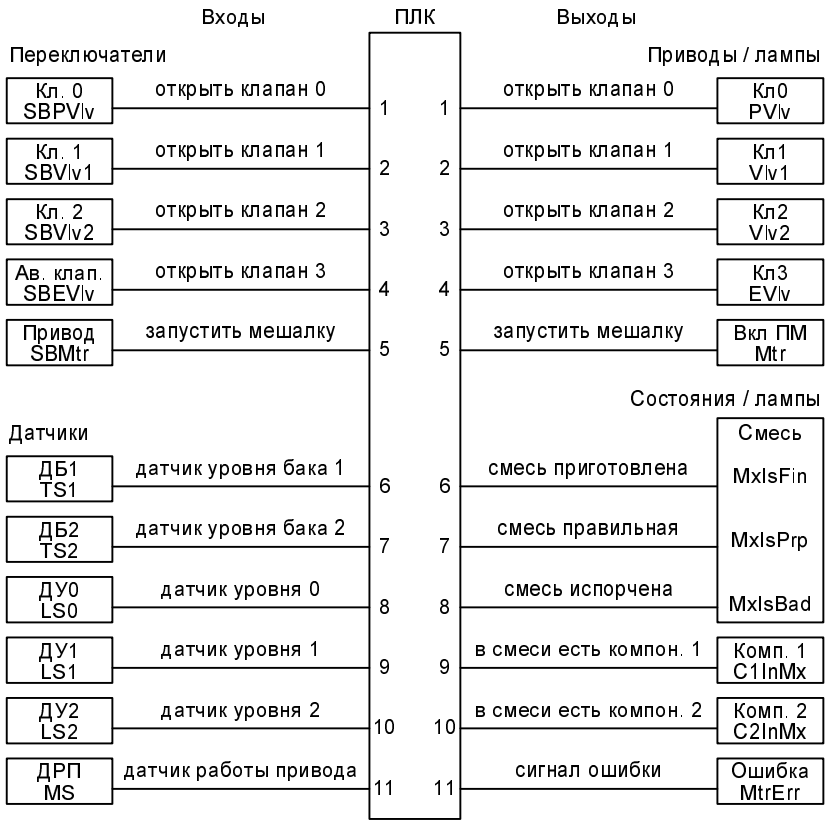

Рис. 2. Интерфейс установки для приготовления смесей

выходной сигнал на лампу «Смесь правильн.»), если резервуар пуст или смесь становится испорченной.

5. Смесь в резервуаре считается приготовленной (выставляется сигнал на лампу «Смесь приготовл.»), если правильная смесь перемешивалась мешалкой непрерывно в течение T секунд. Приготовленная смесь перестает быть таковой (снимается сигнал на лампу «Смесь приготовл.»), если смесь перестает быть правильной.

6. Сигнал ПЛК, открывающий аварийный клапан «Кл3», выставляется и удерживается тогда и только тогда, когда переключатель «Ав. клап.» находится в нажатом состоянии.

7. Клапан «Кл1» может находиться в открытом состоянии (выставлен сигнал открыть клапан 1 и горит лампа «Кл1») только тогда, когда выполнены все следующие условия: 1) нажат переключатель «Кл. 1», 2) бак 1 не пуст, 3) аварийный клапан «Кл3» закрыт, 4) в резервуаре нет второго компонента, 5) уровень 1 в резервуаре не достигнут.

8. Клапан «Кл2» может находиться в открытом состоянии (выставлен сигнал открыть клапан 2 и горит лампа «Кл2») только тогда, когда выполнены все следующие условия: 1) нажат переключатель «Кл. 2», 2) бак 2 не пуст, 3) аварийный клапан «Кл3» закрыт, 4) в резервуаре достигнут уровень 1, но не достигнут уровень 2, 5) смесь не является испорченной и не является приготовленной.

9. Разливной клапан «Кл0» может находиться в открытом состоянии (выставлен

сигнал открыть клапан 0 и горит лампа «Кл0») только тогда, когда выполнены все следующие условия: 1) нажат переключатель «Кл. 0», 2) все другие клапаны закрыты, 3) смесь в резервуаре является приготовленной (т. е. через клапан 0 разрешается сливать только приготовленную смесь).

10. Считается, что в работе привода мешалки произошел сбой (выставляется сигнал на лампу «Ошибка мешалки»), если 1) нажат переключатель «Привод», 2) после включения привода мешалки, т. е. после выставления сигнала «Запустить мешалку», прошло не менее двух секунд, но мешалка так и не заработала (нет сигнала от датчика работы привода «ДРП»), или будучи включенным двигатель мешалки перестал работать. Сигнал об ошибке сбрасывается, как только переключатель «Привод» переходит в отжатое состояние.

11. Запуск привода мешалки, т. е. выставление выходного сигнала «Запустить мешалку» (зажигается лампа «Мешалка включена»), осуществляется тогда и только тогда, когда 1) нажат переключатель «Привод», 2) смесь в резервуаре является правильной, но еще не приготовленной, 3) аварийный клапан «Кл3» закрыт, 4) нет сигнала об ошибке в работе привода мешалки (не горит лампа «Ошибка мешалки»).

12. Отсчет времени запуска и времени работы привода мешалки осуществляется с помощью соответствующих таймеров (специальных программных компонентов).

Программные свойства для тестирования и верификации с учетом согласованного поведения датчиков:

1. «Открытый клапан». Если клапан, исключая аварийный, был открыт, то он обязательно рано или поздно будет закрыт.

2. «Включенный двигатель». Если двигатель мешалки был включен, то он обязательно рано или поздно будет выключен.

3. «Порча смеси». Причиной порчи смеси в резервуаре может быть только открытый аварийный клапан «Кл3».

4. «Перерасход компонентов». В испорченную смесь в резервуаре ни при каких условиях не будут добавляться смешиваемые компоненты из баков.

5. «Некондиционный продукт». Ни при каких условиях испорченная смесь не будет подаваться из резервуара через разливной клапан «Кл0».

6. «Перемешивание». Мешалка будет перемешивать только правильную неиспорченную смесь, которая еще не является полностью приготовленной.

7. «Розлив продукта». Всегда, когда разливной клапан «Кл0» открыт, в резервуаре находится готовая смесь (конечный продукт).

8. «Взаимное исключение». Одновременно в открытом состоянии может находиться не более одного клапана. Не может быть ситуации, при которой мешалка работает и открыт один из клапанов.

9. «Технологический процесс». Если кнопки-переключатели «Кл. 2», «Кл. 1», «Кл. 0» и «Привод» находятся в нажатом положении, а резервуар для смешивания пуст, то при условии, что мешалка не будет ломаться, смешиваемых компонентов будет в достатке и переключатель «Ав. клап.» не будет нажат, рано или поздно либо готовая смесь будет подаваться из резервуара через разливной клапан «Кл0», либо переключатели (хотя бы один) будут отжаты.

10. «Ошибка мешалки». Если выставлен сигнал «Ошибка мешалки», двигатель мешалки находится в выключенном состоянии.

11. «Испорченная смесь». По выставленным сигналам (горящим лампам) испорченная смесь не может быть ни приготовленной, ни правильной.

12. «Приготовленная смесь». По выставленным сигналам (горящим лампам) приготовленная смесь содержит оба компонента, является правильной и не может быть испорченной.

13. «Правильная смесь». По выставленным сигналам (горящим лампам) правильная смесь содержит оба компонента и не может быть испорченной.

14. «Забытый таймер». Если таймер сработал, то но обязательно будет сброшен (уже на следующем проходе рабочего цикла).

Требования к модели поведения датчиков (разбиты по группам):

ДУ0.1.2. Если датчик нулевого уровня «ДУ0» молчит, следовательно, должны молчать датчики первого и второго уровней «ДУ1» и «ДУ2».

ДУ0.2.1. Если датчик «ДУ0» сработал, значит, клапан «Кл1» находился в открытом состоянии и бак 1 не был пуст или клапан «Кл2» находился в открытом состоянии и бак 2 не был пуст, а датчики уровней «ДУ1» и «ДУ2» молчали и продолжают молчать.

ДУ0.2.2. Если датчик «ДУ0» погас, значит, 1) аварийный клапан «Кл3» или разливной клапан «Кл0» находился в открытом состоянии, 2) датчики уровней «ДУ1» и «ДУ2» молчали и продолжают молчать.

ДУ1.1.1. Если датчик первого уровня «ДУ1» горит, следовательно, должен гореть и датчик нулевого уровня «ДУ0».

ДУ1.1.2. Если датчик первого уровня «ДУ1» молчит, следовательно, должен молчать и датчик второго уровня «ДУ2».

ДУ1.2.1. Если датчик «ДУ1» сработал, значит, 1) клапан «Кл1» находился в открытом состоянии и бак 1 не был пуст или клапан «Кл2» находился в открытом состоянии и бак 2 не был пуст, 2) датчик «ДУ2» молчал и продолжает молчать, 3) датчик «ДУ0» горел и продолжает гореть.

ДУ1.2.2. Если датчик «ДУ1» погас, значит, 1) аварийный клапан «Кл3» или разливной клапан «Кл0» находился в открытом состоянии, 2) датчик «ДУ2» молчал и продолжает молчать, 3) датчик «ДУ0» горел и продолжает гореть.

ДУ2.1.1. Если датчик второго уровня «ДУ2» горит, следовательно, должны гореть датчики нулевого и первого уровней «ДУ0» и «ДУ1».

ДУ2.2.1. Если датчик «ДУ2» сработал, значит, 1) клапан «Кл1» находился в открытом состоянии и бак 1 не был пуст или клапан «Кл2» находился в открытом состоянии и бак 2 не был пуст, 2) датчики уровней «ДУ0» и «ДУ1» горели и продолжают гореть.

ДУ2.2.2. Если датчик «ДУ2» погас, значит, 1) аварийный клапан «Кл3» или разливной клапан «Кл0» находился в открытом состоянии, 2) датчики уровней «ДУ0» и «ДУ1» горели и продолжают гореть.

ДУ0.3.1., ДУ1.3.1., ДУ2.3.1. Если датчик «ДУ0», «ДУ1» или «ДУ2» навсегда остается в горящем состоянии, значит, либо 1) через некоторое время ни аварийный клапан «Кл3», ни разливной клапан «Кл0» никогда более не будут открыты, либо 2) всякий раз, когда аварийный клапан «Кл3» или разливной клапан «Кл0» находится в открытом состоянии, рано или поздно в открытом состоянии будет находиться клапан «Кл1» при непустом баке 1 или клапан «Кл2» при непустом баке 2.

ДУ0.3.2., ДУ1.3.2., ДУ2.3.2. Если датчик «ДУ0», «ДУ1» или «ДУ2» замолкает (гаснет) навсегда, значит, либо 1) через некоторое время ни клапан «Кл1» при непустом баке 1, ни клапан «Кл2» при непустом баке 2 никогда более не будут открыты, либо 2) всякий раз, когда в открытом состоянии находится клапан «Кл1» при непустом баке 1 или клапан «Кл2» при непустом баке 2, рано или поздно в открытом состоянии также будет находиться аварийный клапан «Кл3» или разливной клапан «Кл0».

ДБ1.2.2. Если датчик уровня «ДБ1» погас, значит, клапан «Кл1» находился в открытом состоянии.

ДБ2.2.2. Если датчик уровня «ДБ2» погас, значит, клапан «Кл2» находился в открытом состоянии.

ДРП.2.1. Если датчик «ДРП» сработал, значит, двигатель мешалки находился во включенном состоянии.

ДРП.2.2. Если датчик «ДРП» погас, значит, двигатель мешалки находился в выключенном состоянии.

ДРП.3.1. Если датчик «ДРП» навсегда остается в горящем состоянии, следовательно, двигатель мешалки включается бесконечно часто.

Ниже приводится ПЛК-программа управления установкой, построенная на стандартном языке ST по заданному описанию, и ее SMV-модель с представлением программных свойств и требований к согласованному поведению датчиков в виде LTL-формул. Проверка соответствия модели программным свойствам с учетом согласованного поведения датчиков осуществляется автоматически с помощью верификатора Cadence SMV [10]. Изначально в SMV-модели для каждого датчика S рассматриваются все варианты (включая несогласованные) его входных значений через конструкцию недетерминизма $\mathrm{next}(S){:=}\{0,1\}$. Далее при доказательстве соответствия модели программным свойствам используется конструкция using...prove. В разделе using перечисляются LTL-требования согласованности всех датчиков, которые учитываются при доказательстве справедливости свойств, перечисленных в разделе prove.

Программа ПЛК управления установкой для приготовления смесей (язык ST).

```
VAR_GLOBAL                                (* 11 входов  *)
  SBVlv1, SBVlv2, SBEVlv, SBPVlv, SBMtr:  BOOL;           (* кнопки-переключатели *)
  TS1, TS2, LS0, LS1, LS2, MS:  BOOL;                     (* датчики              *)
  Vlv1, Vlv2, EVlv, PVlv, Mtr:  BOOL;  (* 11 выходов *)  (* приводы/лампы        *)
  MxIsFin, MxIsBad, MxIsPrp, C1InMx, C2InMx, MtrErr: BOOL; (* состояния/лампы     *)
END_VAR (* по умолчанию переменные инициализируются нулем *)
PROGRAM PLC_PRG
VAR
  ErrTmr: TON := (PT := T#2s);
  MtrTmr: TON := (PT := T#7s);
  (* вспомогательные псевдооператорные переменные *)
  _C1InMx, _C2InMx, _MtrErr, _MxIsFin, _MxIsBad, _MxIsPrp       : BOOL;
  _Vlv1, _Vlv2, _EVlv, _PVlv, _Mtr, _TS1, _TS2, _MS, _LS1, _LS2 : BOOL;
END_VAR
MtrTmr.In := _Mtr AND     MS;  MtrTmr();
ErrTmr.In := _Mtr AND NOT MS;  ErrTmr();
IF NOT _C1InMx AND     LS0 AND _Vlv1 AND _TS1  THEN C1InMx:=1;
ELSIF  _C1InMx AND NOT LS0                     THEN C1InMx:=0; END_IF;
```

```
IF NOT _C2InMx AND     LS0 AND _Vlv2 AND _TS2  THEN C2InMx:=1;
ELSIF  _C2InMx AND NOT LS0                     THEN C2InMx:=0; END_IF;
IF NOT _MxIsBad AND LS0 AND (    _MxIsPrp AND (_Vlv1 AND _TS1 OR _Vlv2 AND _TS2) OR
                             NOT _MxIsPrp AND _C2InMx AND  (_EVlv OR _PVlv)      OR
                             (NOT _LS1 OR _LS2) AND _Vlv2 AND _TS2               OR
                                         _LS1  AND _Vlv1 AND _TS1                 )
                                                                  THEN MxIsBad:=1;
ELSIF  _MxIsBad AND (NOT LS0 OR NOT LS1 AND NOT C2InMx)           THEN MxIsBad:=0;
END_IF;
IF NOT _MxIsPrp AND      LS2 AND NOT MxIsBad  THEN MxIsPrp:=1;
ELSIF  _MxIsPrp AND (NOT LS0 OR      MxIsBad) THEN MxIsPrp:=0; END_IF;
IF NOT _MxIsFin AND MtrTmr.Q AND     MxIsPrp  THEN MxIsFin:=1;
ELSIF  _MxIsFin AND              NOT MxIsPrp  THEN MxIsFin:=0; END_IF;
EVlv := SBEVlv;
Vlv1 := SBVlv1 AND TS1 AND NOT EVlv AND NOT LS1 AND NOT C2InMx;
Vlv2 := SBVlv2 AND TS2 AND NOT EVlv AND LS1 AND NOT LS2 AND NOT(MxIsPrp OR MxIsBad);
PVlv := SBPVlv AND NOT(EVlv OR Vlv1 OR Vlv2) AND MxIsFin;
IF NOT _MtrErr AND SBMtr AND (ErrTmr.Q OR _Mtr AND _MS) AND NOT MS THEN MtrErr:=1;
ELSIF  _MtrErr AND NOT SBMtr                                      THEN MtrErr:=0;
END_IF;
Mtr := SBMtr AND MxIsPrp AND NOT MxIsFin AND NOT MtrErr AND NOT EVlv;
(* псевдооператорный раздел *)
_MtrErr:=MtrErr; _C1InMx:=C1InMx; _C2InMx:=C2InMx; _MxIsFin:=MxIsFin;
_MxIsBad:=MxIsBad; _MxIsPrp:=MxIsPrp; _LS1:=LS1; _LS2:=LS2; _Vlv1:=Vlv1;
_Vlv2:=Vlv2; _EVlv:=EVlv; _PVlv:=PVlv; _Mtr:=Mtr; _TS1:=TS1; _TS2:=TS2; _MS :=MS;
```

SMV-модель программы управления установкой для приготовления смесей.

```
module timer(){
  I : 0..1;   /* вход  */
  Q : 0..1;   /* выход */
  init(I):=0; init(Q):=0; /* инициализация */
  next(Q):= next(I) & (Q | {0, 1}); /* новое значение выхода */
  FAIRNESS I -> Q;  /* условие честного срабатывания таймера */
}
module main() {  /* Раздел описания переменных */
  SBVlv1, SBVlv2, SBEVlv, SBPVlv, SBMtr: 0..1;             /* Кнопки-переключатели */
  TS1, TS2, LS0, LS1, LS2, MS:           0..1;                  /* Датчики         */
  Vlv1, Vlv2, EVlv, PVlv, Mtr:           0..1;                  /* Приводы/лампы   */
  MxIsFin, MxIsBad, MxIsPrp, C1InMx, C2InMx, MtrErr: 0..1;      /* Состояния/лампы */
  ErrTmr, MtrTmr : timer;                                       /* Таймеры         */
  /* Раздел инициализации */
  init(MS):=0; init(TS1):=0; init(TS2):=0; init(LS0):=0; init(LS1):=0; init(LS2):=0;
  init(SBVlv1):=0; init(SBVlv2):=0; init(SBEVlv):=0; init(SBPVlv):=0; init(SBMtr):=0;
  init(Vlv1):=0; init(Vlv2):=0; init(EVlv):=0; init(PVlv):=0; init(Mtr):=0;
  init(MxIsFin):=0; init(MxIsBad):=0; init(MxIsPrp):=0; init(MtrErr):=0;
  init(C1InMx):=0; init(C2InMx):=0;
  /* Система переходов */   /* Входы */
  next(SBVlv1):={0,1}; next(SBVlv2):={0,1}; next(SBEVlv):={0,1}; next(SBPVlv):={0,1};
  next(SBMtr) :={0,1};                                          /* Приводы/лампы */
  next(MS) :={0,1}; next(TS1):={0,1}; next(TS2):={0,1};
  next(LS0):={0,1}; next(LS1):={0,1}; next(LS2):={0,1};                 /* Датчики */
                           /* Выходы, таймеры и внутренние переменные */
  next(MtrTmr.I) :=  next(MS) & Mtr;
  next(ErrTmr.I) := ~next(MS) & Mtr;
  case{ ~C1InMx &  next(LS0) & Vlv1 & TS1 : next(C1InMx) := 1;
```

```
       C1InMx & ~next(LS0)              : next(C1InMx) := 0;
      default                           : next(C1InMx) := C1InMx; };
case{ ~C2InMx &  next(LS0) & Vlv2 & TS2 : next(C2InMx) := 1;
       C2InMx & ~next(LS0)              : next(C2InMx) := 0;
      default                           : next(C2InMx) := C2InMx; };
case{~MxIsBad & next(LS0) & ( MxIsPrp & (Vlv1 & TS1 | Vlv2 & TS2) |
                             ~MxIsPrp & C2InMx &  (EVlv | PVlv)   |
                             (~LS1 | LS2) & Vlv2 & TS2            |
                                    LS1  & Vlv1 & TS1)  : next(MxIsBad):=1;
     MxIsBad & (~next(LS0) | ~next(LS1) & ~next(C2InMx)): next(MxIsBad):=0;
     default                                            : next(MxIsBad):=MxIsBad;};
case{ ~MxIsPrp &   next(LS2) & ~next(MxIsBad)  : next(MxIsPrp) := 1;
       MxIsPrp & (~next(LS0) |  next(MxIsBad)) : next(MxIsPrp) := 0;
      default                                  : next(MxIsPrp) := MxIsPrp; };
case{ ~MxIsFin & next(MtrTmr.Q) &  next(MxIsPrp) : next(MxIsFin) := 1;
       MxIsFin &                  ~next(MxIsPrp) : next(MxIsFin) := 0;
      default                                    : next(MxIsFin) := MxIsFin; };
next(EVlv):= next(SBEVlv);
next(Vlv1):= next(SBVlv1) & next(TS1) & ~next(EVlv) & ~next(LS1) & ~next(C2InMx);
next(Vlv2):= next(SBVlv2) & next(TS2) & ~next(EVlv) &
             next(LS1) & ~next(LS2) & ~(next(MxIsPrp) | next(MxIsBad));
next(PVlv):=next(SBPVlv) & ~(next(EVlv) | next(Vlv1) | next(Vlv2)) & next(MxIsFin);
case{ ~MtrErr &  next(SBMtr) &
                 next(ErrTmr.Q) | Mtr & MS) & ~next(MS) : next(MtrErr) := 1;
       MtrErr & ~next(SBMtr)                            : next(MtrErr) := 0;
      default                                           : next(MtrErr) := MtrErr;};
next(Mtr):= next(SBMtr) & next(MxIsPrp) & ~next(MxIsFin) &
                                                  ~next(MtrErr) & ~next(EVlv);
/* Требования согласованной работы датчиков  */
TS1_2_2: assert G( TS1 & ~X(TS1) -> Vlv1 );                          /* ДБ1.2.2 */
TS2_2_2: assert G( TS2 & ~X(TS2) -> Vlv2 );                          /* ДБ2.2.2 */
MS_2_1:  assert G( ~MS &  X(MS)  ->  Mtr );                          /* ДРП.2.1 */
MS_2_2:  assert G(  MS & ~X(MS)  -> ~Mtr );                          /* ДРП.2.2 */
MS_3_1:  assert G( G(MS) -> G(F(Mtr)) );                             /* ДРП.3.1 */
LS0_1_2: assert G(        ~X(LS0) ->  ~X(LS1) & ~X(LS2) );           /* ДУ0.1.2 */
LS0_2_1: assert G( ~LS0 &  X(LS0) ->  ~LS1 & ~LS2 & ~X(LS1) & ~X(LS2) &
                                      (Vlv1 & TS1 | Vlv2 & TS2) );   /* ДУ0.2.1 */
LS0_2_2: assert G(  LS0 & ~X(LS0) ->  ~LS1 & ~LS2 & ~X(LS1) & ~X(LS2) &
                                      (EVlv | PVlv) );               /* ДУ0.2.2 */
LS0_3_1: assert G( G( LS0) ->  G((EVlv | PVlv) -> F(Vlv1 & TS1 | Vlv2 & TS2)) |
                               F(G~(EVlv | PVlv)) );                 /* ДУ0.3.1 */
LS0_3_2: assert G( G(~LS0) ->  G((Vlv1 & TS1 | Vlv2 & TS2) -> F(EVlv | PVlv)) |
                               F(G~(Vlv1 & TS1 | Vlv2 & TS2)) );     /* ДУ0.3.2 */
LS1_1_1: assert G(         X(LS1) ->  X(LS0));                       /* ДУ1.1.1 */
LS1_1_2: assert G(        ~X(LS1) -> ~X(LS2));                       /* ДУ1.1.2 */
LS1_2_1: assert G( ~LS1 &  X(LS1) ->  LS0 & X(LS0) & ~LS2 & ~X(LS2) &
                                      (Vlv1 & TS1 | Vlv2 & TS2) );   /* ДУ1.2.1 */
LS1_2_2: assert G(  LS1 & ~X(LS1) ->  LS0 & X(LS0) & ~LS2 & ~X(LS2) &
                                      (EVlv | PVlv) );               /* ДУ1.2.2 */
LS1_3_1: assert G( G( LS1) ->  G((EVlv | PVlv) -> F(Vlv1 & TS1 | Vlv2 & TS2)) |
                               F(G~(EVlv | PVlv)) );                 /* ДУ1.3.1 */
LS1_3_2: assert G( G(~LS1) ->  G((Vlv1 & TS1 | Vlv2 & TS2) -> F(EVlv | PVlv)) |
                               F(G~(Vlv1 & TS1 | Vlv2 & TS2)) );     /* ДУ1.3.2 */
LS2_1_1: assert G(         X(LS2) ->  X(LS1) & X(LS0));              /* ДУ2.1.1 */
LS2_2_1: assert G( ~LS2 &  X(LS2) ->  LS1 & LS0 & X(LS1) & X(LS0) &
                                      (Vlv1 & TS1 | Vlv2 & TS2) );   /* ДУ2.2.1 */
```

```
  LS2_2_2: assert G(  LS2 & ~X(LS2) ->  LS1 & LS0 & X(LS1) & X(LS0) &
                                        (EVlv | PVlv) );                /* ДУ2.2.2 */
  LS2_3_1: assert G( G( LS2) ->  G((EVlv | PVlv) -> F(Vlv1 & TS1 | Vlv2 & TS2)) |
                                 F(G~(EVlv | PVlv)) );                  /* ДУ2.3.1 */
  LS2_3_2: assert G( G(~LS2) ->  G((Vlv1 & TS1 | Vlv2 & TS2) -> F(EVlv | PVlv)) |
                                 F(G~(Vlv1 & TS1 | Vlv2 & TS2)) );      /* ДУ2.3.2 */
  /* Раздел свойств */
  Prp_Vlv1: assert G(Vlv1 -> F(~Vlv1));              /* свойство "Открытый клапан" */
  Prp_Vlv2: assert G(Vlv2 -> F(~Vlv2));
  Prp_PVlv: assert G(PVlv -> F(~PVlv));
  Prp_Mtr:  assert G(Mtr  -> F(~Mtr));                 /*   "Включенный двигатель" */
  Prp_EVlv: assert G(~MxIsBad & X(MxIsBad) -> EVlv);   /*            "Порча смеси" */
  Prp_MxIsBad_1: assert G(MxIsBad -> ~Vlv1 & ~Vlv2);   /* "Перерасход компонентов" */
  Prp_MxIsBad_2: assert G(MxIsBad -> ~PVlv);           /* "Некондиционный продукт" */
  Prp_Mxng: assert G(Mtr -> MxIsPrp & ~MxIsBad & ~MxIsFin); /*     "Перемешивание" */
  Prp_FinPVlv: assert G(PVlv -> MxIsFin);              /*        "Розлив продукта" */
  Prp_Vlvs: assert G(EVlv+PVlv+Vlv1+Vlv2+Mtr <=1);     /*    "Взаимное исключение" */
  Prp_Proc: assert G(SBVlv1 & SBVlv2 & SBPVlv & SBMtr & ~LS0 ->    /* "Технологич. */
            F(MxIsFin & PVlv | ~(SBVlv1 & SBVlv2 & SBPVlv & SBMtr))); /*  процесс" */
  Prp_MtrErr: assert G(MtrErr -> ~Mtr);                /*         "Ошибка мешалки" */
  Prp_MxIsBad_3: assert G(MxIsBad -> ~MxIsFin & ~MxIsPrp); /*  "Испорченная смесь" */
  Prp_MxIsFin: assert G(MxIsFin ->
                  MxIsPrp & ~MxIsBad & C1InMx & C2InMx); /* "Приготовленная смесь" */
  Prp_MxIsPrp: assert G(MxIsPrp -> ~MxIsBad & C1InMx & C2InMx);/*"Правильная смесь"*/
  Prp_ErrTmr: assert G(ErrTmr.Q -> X(~ErrTmr.Q));
  Prp_MtrTmr: assert G(MtrTmr.Q -> X(~MtrTmr.Q));      /*         "Забытый таймер" */

  /* Проверка свойств с учетом согласованного поведения датчиков */
using TS1_2_2, TS2_2_2, MS_2_1, MS_2_2, MS_3_1, LS1_1_1, LS1_1_2, LS0_1_2, LS0_2_1,
      LS0_2_2, LS0_3_1, LS0_3_2, LS1_2_1, LS1_2_2, LS1_3_1, LS1_3_2, LS2_1_1,
      LS2_2_1, LS2_2_2, LS2_3_1, LS2_3_2
prove Prp_Vlv1, Prp_Vlv2, Prp_PVlv, Prp_Mtr, Prp_EVlv, Prp_MxIsBad_1, Prp_MxIsBad_2,
      Prp_Mxng, Prp_FinPVlv, Prp_Vlvs, Prp_Proc, Prp_MtrErr, Prp_MxIsBad_3,
      Prp_MxIsFin, Prp_MxIsPrp, Prp_ErrTmr, Prp_MtrTmr;
  /* Дополнительные условия для проверки свойства Prp_Proc */
  Cnd_Comp1:  assert G( F(TS1) );
  Cnd_Comp2:  assert G( F(TS2) );
  Cnd_SBEVlv: assert G( (SBVlv1 | SBVlv2 | SBPVlv | SBMtr) -> ~SBEVlv );
  Cnd_SBMtr:  assert G( Mtr -> (Mtr U MS) & (MS -> X(MS)) );
using Cnd_Comp1, Cnd_Comp2, Cnd_SBEVlv, Cnd_SBMtr
prove Prp_Proc;
}
```

Заключение

Программа ПЛК представляет собой описание (с использованием различных языков программирования) реакций на входные сигналы от датчиков, переключателей и кнопок. Предложенный в статье подход к моделированию согласованного поведения ПЛК-датчиков при построении модели программы ПЛК позволяет сосредоточиться на моделировании именно этих реакций по тексту программы без внедрения в код модели (например в SMV-код) дополнительных конструкций, призванных отразить реалистичное поведение датчиков. Согласованное поведение датчиков учитывается лишь на стадии проверки соответствия программной модели требуе-

мым свойствам, т. е. доказательство выполнимости свойств для построенной модели происходит с условием, что рассматриваемая модель содержит только те исполнения исходной программы, которые отвечают требованиям согласованного поведения датчиков.

Список литературы

1. *Рябухин Д. А.*, *Кузьмин Е. В.*, *Соколов В. А.* Построение IL-программ ПЛК по LTL-спецификации // Моделирование и анализ информационных систем. 2014. Т. 21, №2. С. 26–38.
(*Ryabukhin D. A.*, *Kuzmin E. V.*, *Sokolov V. A.* Construction of PLC IL-programs by LTL-specification // Modeling and analysis of information systems. 2014. V. 21, №2. P. 26–38 [in Russian]).

2. *Кузьмин Е. В.*, *Соколов В. А.*, *Рябухин Д. А.* Построение и верификация LD-программ ПЛК по LTL-спецификации // Моделирование и анализ информационных систем. 2013. Т. 20, №6. С. 78–94.
(*Kuzmin E. V.*, *Sokolov V. A.*, *Ryabukhin D. A.* Construction and Verification of PLC LD-programs by LTL-specification // Modeling and analysis of information systems. 2013. V. 20, №6. P. 78–94 [in Russian]).

3. *Кузьмин Е. В.*, *Соколов В. А.*, *Рябухин Д. А.* Построение и верификация ПЛК-программ по LTL-спецификации // Моделирование и анализ информационных систем. 2013. Т. 20, №4. С. 5–22. (*Kuzmin E. V.*, *Sokolov V. A.*, *Ryabukhin D. A.* Construction and Verification of PLC-programs by LTL-specification // Modeling and analysis of information systems. 2013. V. 20, №4. P. 5–22 [in Russian]).

4. *Кузьмин Е. В.*, *Рябухин Д. А.*, *Шипов А. А.* Построение и верификация ПЛК-программ по LTL-спецификации // Международная научно-практическая конференция «Инструменты и методы анализа программ». Кострома, 2013. С. 17–34.
(*Kuzmin E. V.*, *Ryabukhin D. A.*, *Shipov A. A.* Construction and Verification of PLC-programs by LTL-specification // Proc. of Int. Conf. «Tools and Methods of Program Analysis (TMPA-2013)». Kostroma, 2013. P. 17–34 [in Russian]).

5. *Кузьмин Е. В.*, *Соколов В. А.* Моделирование, спецификация и построение программ логических контроллеров // Моделирование и анализ информационных систем. 2013. Т. 20, №2. С. 104–120. (*Kuzmin E. V.*, *Sokolov V. A.* Modeling, Specification and Construction of PLC-programs // Modeling and analysis of information systems. 2013. V. 20, №2. P. 104–120 [in Russian]).

6. *Петров И. В.* Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. М.: СОЛОН-Пресс, 2004. 256 с. [*Petrov I. V.* Programmiruemye kontrollery. Standartnye jazyki i priemy prikladnogo proektirovanija. M.: SOLON-Press, 2004. 256 p. (in Russian)].

7. *Clark E. M.*, *Grumberg O.*, *Peled D. A.* Model Checking. The MIT Press, 2001.

8. CoDeSys. Controller Development System. http://www.3s-software.com/

9. *Parr E. A.* Programmable Controllers. An engineer's guide. Newnes, 2003. 442 p.

10. SMV. The Cadence SMV Model Checker. http://www.kenmcmil.com/smv.html

Modeling a Consistent Behavior of PLC-Sensors

Kuzmin E. V., Ryabukhin D. A., Sokolov V. A.

P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia

Keywords: programmable logic controllers, software engineering, specification and verification of PLC-programs, sensor modeling

The article extends the cycle of papers dedicated to programming and verificatoin of PLC-programs by LTL-specification. This approach provides the availability of correctness analysis of PLC-programs by the model checking method.

The model checking method needs to construct a finite model of a PLC program. For successful verification of required properties it is important to take into consideration that not all combinations of input signals from the sensors can occur while PLC works with a control object. This fact requires more advertence to the construction of the PLC-program model.

In this paper we propose to describe a consistent behavior of sensors by three groups of LTL-formulas. They will affect the program model, approximating it to the actual behavior of the PLC program. The idea of LTL-requirements is shown by an example.

A PLC program is a description of reactions on input signals from sensors, switches and buttons. In constructing a PLC-program model, the approach to modeling a consistent behavior of PLC sensors allows to focus on modeling precisely these reactions without an extension of the program model by additional structures for realization of a realistic behavior of sensors. The consistent behavior of sensors is taken into account only at the stage of checking a conformity of the programming model to required properties, i. e. a property satisfaction proof for the constructed model occurs with the condition that the model contains only such executions of the program that comply with the consistent behavior of sensors.

Сведения об авторах:
Кузьмин Егор Владимирович,
Ярославский государственный университет им. П.Г. Демидова,
д-р физ.-мат. наук, профессор;
Рябухин Дмитрий Александрович,
Ярославский государственный университет им. П.Г. Демидова,
аспирант;
Соколов Валерий Анатольевич,
Ярославский государственный университет им. П.Г. Демидова,
д-р физ.-мат. наук, профессор.