Оценка алгоритмов планирования набора задач в модели системы реального времени методом Model Checking

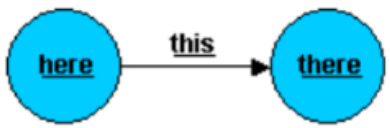
Старолетов Сергей

serg_soft@mail.ru

OSDAY 2021, 15 октября 2021 г.

Необходимость в моделировании

- Абстракция от аппаратной платформы
- Абстракция от языков программирования / компиляторов



Моделирование операционных систем

seL4[1]

- Выполнимая спецификация на функциональном языке
- Доказательства корректности методом Theorem Proving
- Возможности преобразовывать одно в другое

[1] https://github.com/seL4/l4v

ОС реального времени

- Управляют кибер-физическими системами, носимыми устройствами и тд.
- Работают на маленьких энергоэффективных чипах
- Жесткие требования к скорости работы и объему кода и памяти

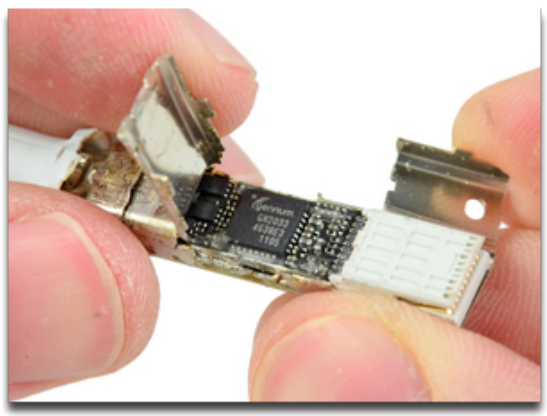
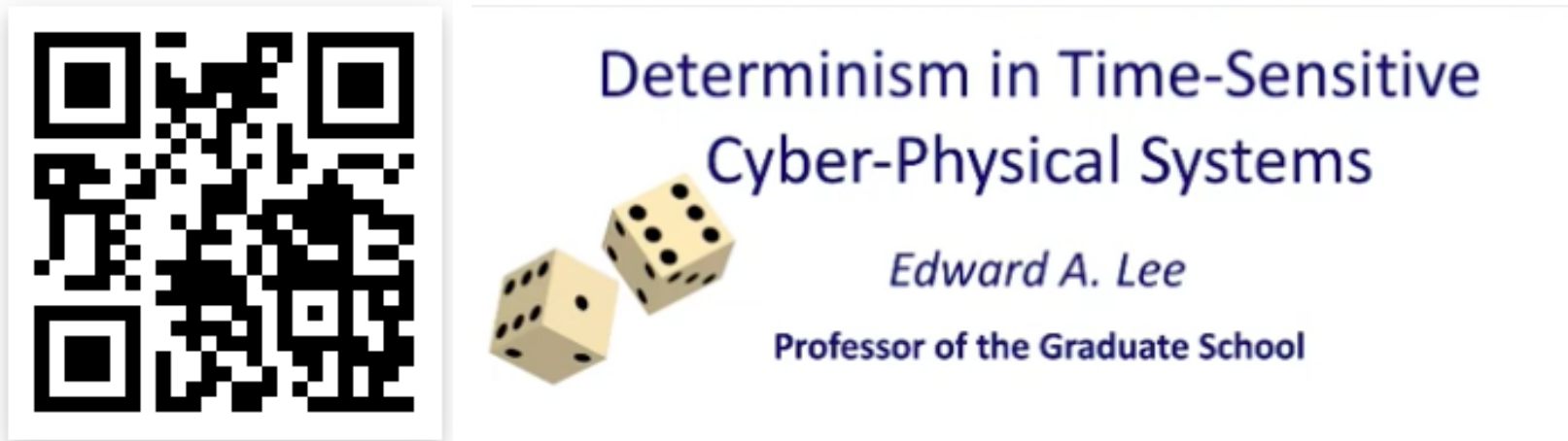


Рис.: Inside Apple's Thunderbolt cable[2]

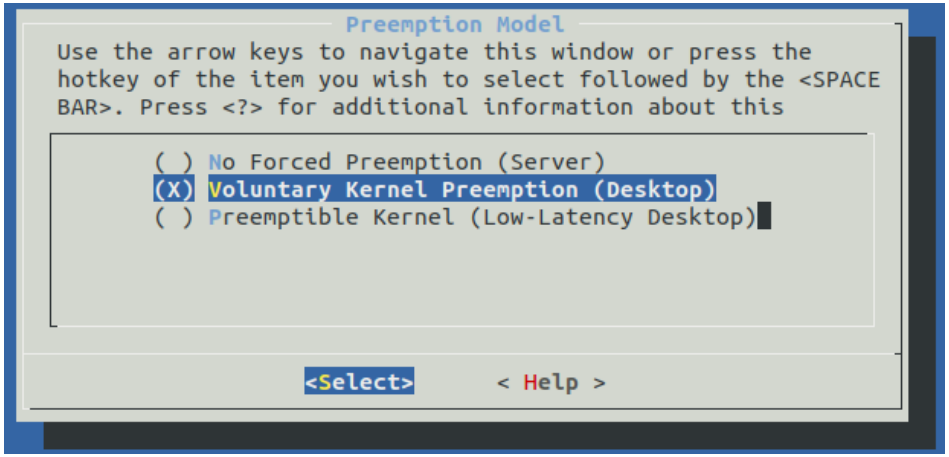
[2] https://www.macobserver.com/tmo/article/apples_thunderbolt_cable_its_full_of_chips

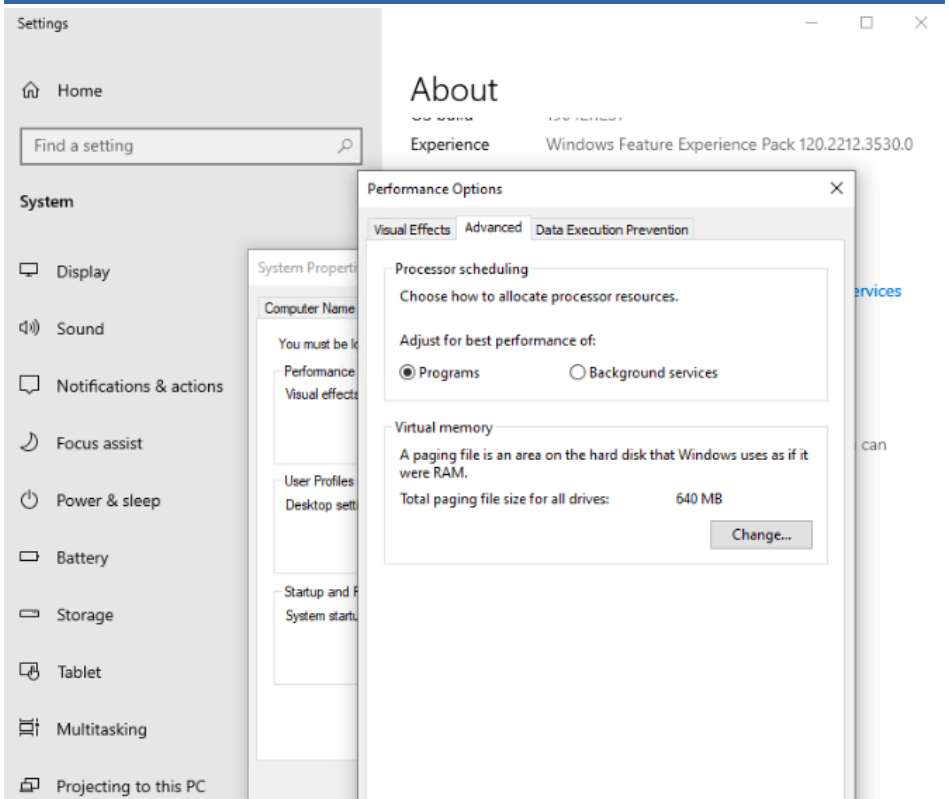
Реальное время

Edward A Lee. Determinism in Time Sensitive Cyber-Physical Systems



Проблема планирования задач





Пример реальной киберфизической системы (ArduPilot)

```
777 static const AP_Scheduler::Task scheduler_tasks[] PROGMEM = {
778     { rc_loop,               4,     10 },
779     { throttle_loop,         8,     45 },
780     { update_GPS,            8,     90 },
781     { update_batt_compass,  40,     72 },
782     { read_aux_switches,    40,      5 },
783     { arm_motors_check,     40,      1 },
784     { auto_trim,            40,     14 },
785     { update_altitude,      40,    100 },
786     { run_nav_updates,       8,     80 },
787     { update_thr_cruise,    40,     10 },
788     { three_hz_loop,       133,      9 },
789     { compass_accumulate,    8,     42 },
790     { barometer_accumulate,  8,     25 },
791 #if FRAME_CONFIG == HELI_FRAME
792     { check_dynamic_flight,  8,     10 },
793 #endif
794     { update_notify,         8,     10 },
795     { one_hz_loop,         400,     42 },
796     { ekf_dcm_check,        40,      2 },
797     { crash_check,          40,      2 },
798     { gcs_check_input,       8,    550 },
799     { gcs_send_heartbeat,  400,    150 },
800     { gcs_send_deferred,     8,    720 },
801     { gcs_data_stream_send,  8,    950 },
802 #if COPTER_LEDS == ENABLED
803     { update_copter_leds,   40,      5 },
804 #endif
805     { update_mount,          8,     45 },
806     { ten_hz_logging_loop,  40,     30 },
807     { fifty_hz_logging_loop, 8,     22 },
808     { perf_update,        4000,     20 },
809     { read_receiver_rssi,   40,      5 },
810 #if FRSKY_TELEM_ENABLED == ENABLED
811     { telemetry_send,       80,     10 },
812 #endif
```

Пример реальной киберфизической системы (открытый код ArduPilot)

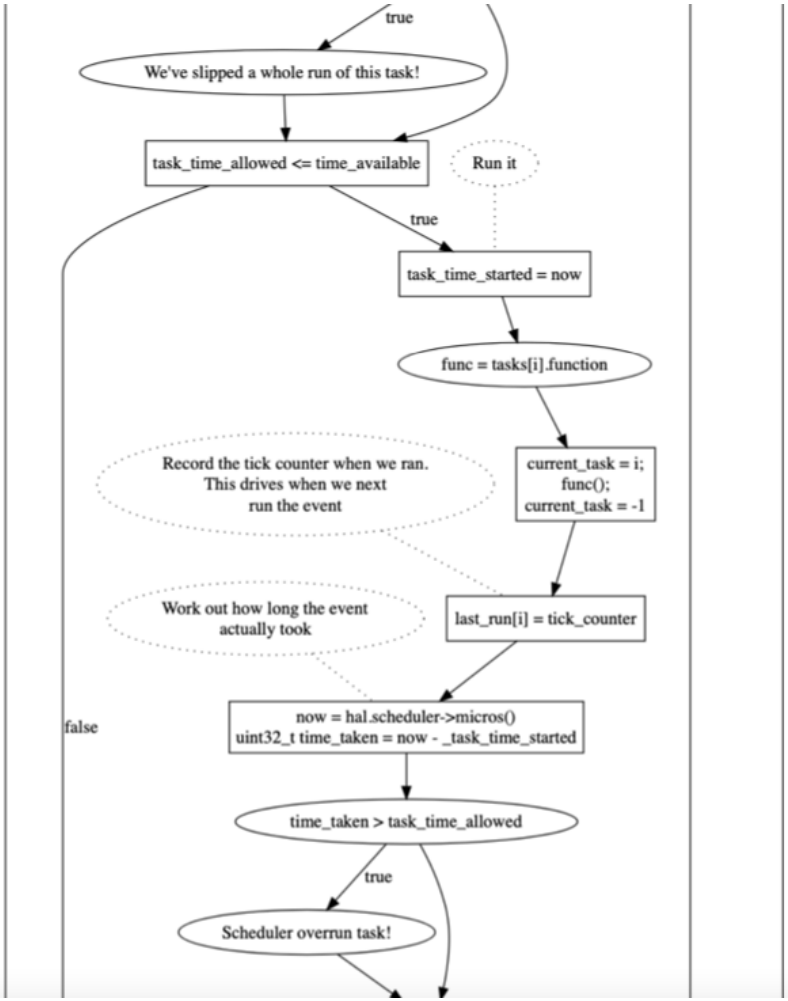
	Task id	Interval ticks	Max time (ms)	Description
1	rc_loop	1	100	Reads user input from transmitter/receiver
2	throttle_loop	2	450	Throttle loop: 1) gets altitude and climb rate from inertial lib; 2) checks if we have landed; 3) checks auto_armed status
3	update_GPS	2	900	GPS data obtaining, logging and glitch protection run after every GPS message, check for loss of GPS
4	update_batt_compass	10	720	Reads battery and compass, records throttle output
5	read_aux_switches	10	50	Checks aux switch positions and invokes configured actions
6	arm_motors_check	10	10	Checks for pilot input to arm or disarm the copter
7	auto_trim	10	140	Slightly adjusts the ahrs.roll˙trim and ahrs.pitch˙trim towards the current stick positions. Meant to be called continuously while the pilot attempts to keep the copter level
8	update_altitude	10	1000	Reads barometer and sonar altitude
9	run_nav_updates	4	800	Top level call for the autopilot. Ensures calculations such as ”distance to waypoint” are calculated before autopilot makes decisions. 1) fetches position from inertial navigation; 2) calculates distance and bearing for reporting and autopilot decisions; 3) runs autopilot to make high level decisions about control modes
10	update_thr_cruise	1	50	Updates throttle cruise if necessary: 1) gets throttle output; 2) calculates average throttle if we are in a level hover 3) updates position controller
11	three_hz_loop	33	90	3.3hz loop: 1) checks if we have lost contact with the ground station; 2) checks if we have breached a fence; 3) updates events
12	compass_accumulate	2	420	If the compass is enabled then tries to accumulate a reading
13	barometer_accumulate	2	250	Tries to accumulate a barometer reading
14	update_notify	2	100	Updates the status of notify (LEDs)
15	one_hz_loop	100	420	Runs at 1Hz: 1) logs battery info to the dataflash; 2) performs pre-arm checks and display failures every 30 seconds; 3) auto disarm checks; 4) makes it possible to change AHRS orientation at runtime during initial config; 5) checks the user has not updated the frame orientation; 6) updates assigned functions and enable auxiliar servos

Пример реальной киберфизической системы (открытый код ArduPilot)

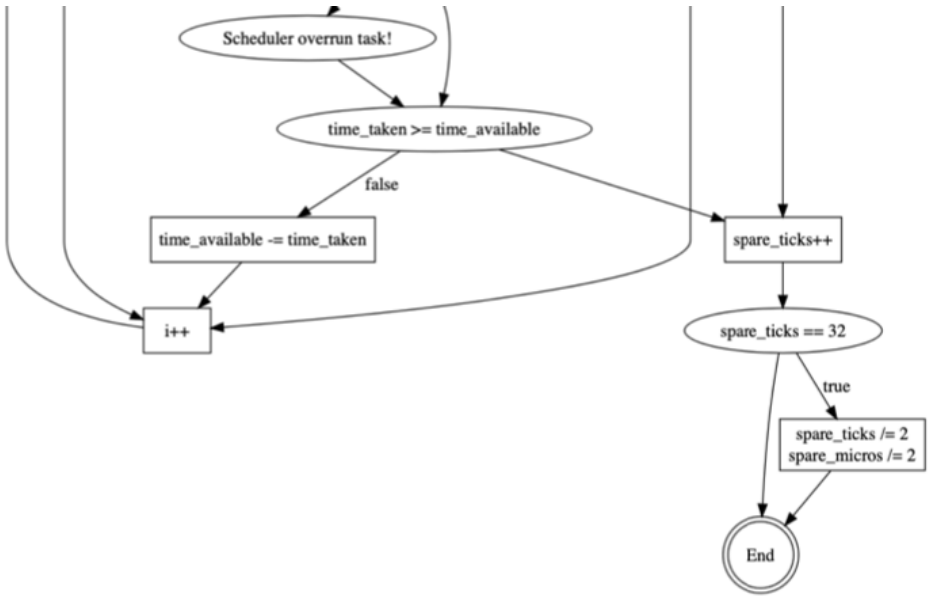
16	ekf_dcm_check	10	20	Detects if EKF (the position and attitude estimation system) variances or DCM yaw errors that are out of tolerance and triggers failsafe
17	crash_check	10	20	Disarms motors if a crash has been detected. Crashes are detected by the vehicle being more than 20 degrees beyond its angle limits continuously for more than 1 second
18	gcs_check_input	2	550	Looks for incoming commands on the GCS links
19	gcs_send_heartbeat	100	150	GCS send message(MSG_HEARTBEAT)
20	gcs_send_deferred	2	720	GCS send message(MSG_RETRY_DEFERRED)
21	gcs_data_stream_send	2	950	Sends data streams in the given rate range on both links
22	update_mount	2	450	Updates camera mount position
23	ten_hz_logging_loop	10	300	Logging
24	fifty_hz_logging_loop	2	220	Logging
25	perf_update	1000	200	Logging performance
26	read_receiver_rssi	10	50	Reads the receiver RSSI as an 8 bit number for MAVLink RC_CHANNELS_SCALED message
27	telemetry_send	20	100	Sends FrSky telemetry if enabled
28	userhook_fast_loop	1	100	Runs 100Hz user hook if enabled
29	userhook_50_hz	2	100	Runs 50Hz user hook if enabled
30	userhook_medium_loop	10	100	Runs 10Hz user hook if enabled
31	userhook_slow_loop	30	100	Runs 3Hz user hook if enabled
32	userhook_super_slow_loop	100	100	Runs 1Hz user hook if enabled

Пример реальной киберфизической системы (открытый код ArduPilot)

Пример реальной киберфизической системы (открытый код ArduPilot)



Пример реальной киберфизической системы (открытый код ArduPilot)[3]



[3]Staroletov, Sergey. "Work-in-Progress Abstract: Revealing and Analyzing Architectural Models in Open-source ArduPilot."2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE, 2021.

Алгоритмы планирования задач в ОС РВ

- На множестве готовых к выполнению задач принимают решение, какая задача должна выполняться в заданный момент времени
- Используют предопределенные параметры задачи, а также производные параметры, вычисленные в текущий момент

Подразделяются по способу анализа исходных данных на

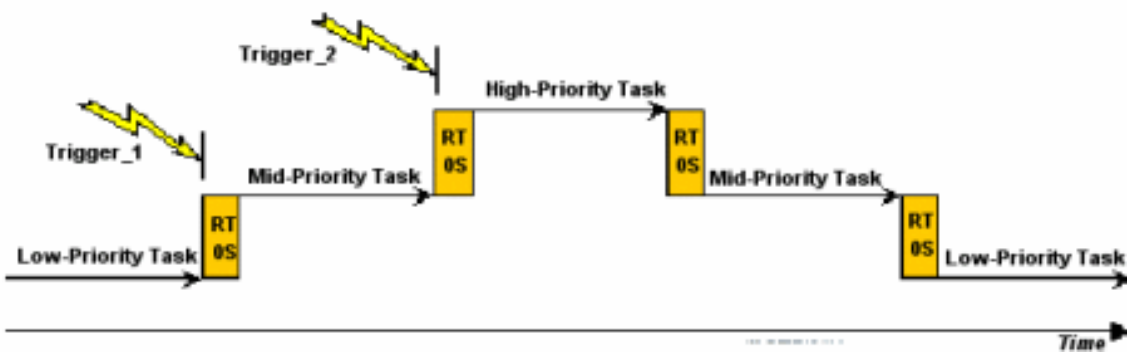
- Статические
- Динамические

Подразделяются по типу вытеснения низкоприоритеных задач высокоприоритетными

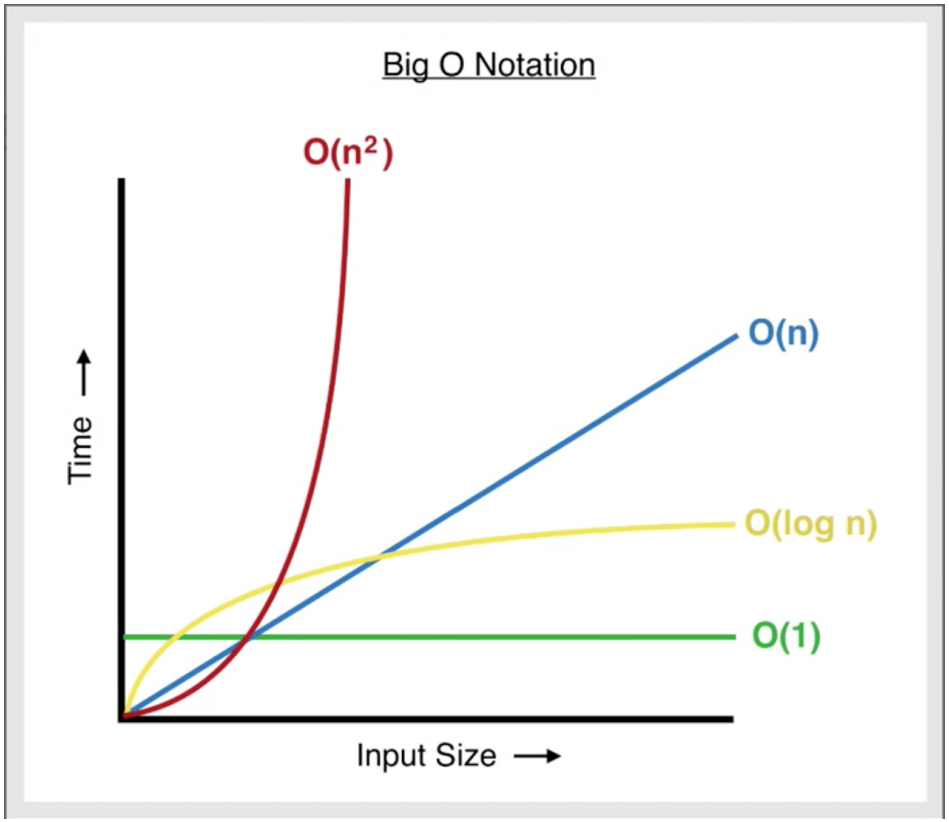
- Вытесняющие
- Невытесняющие

Алгоритмы планирования задач в ОС РВ

Вытеснение задач (D. Kalinsky. Basic concepts of real-time operating systems)



Алгоритмы планирования задач в ОС РВ

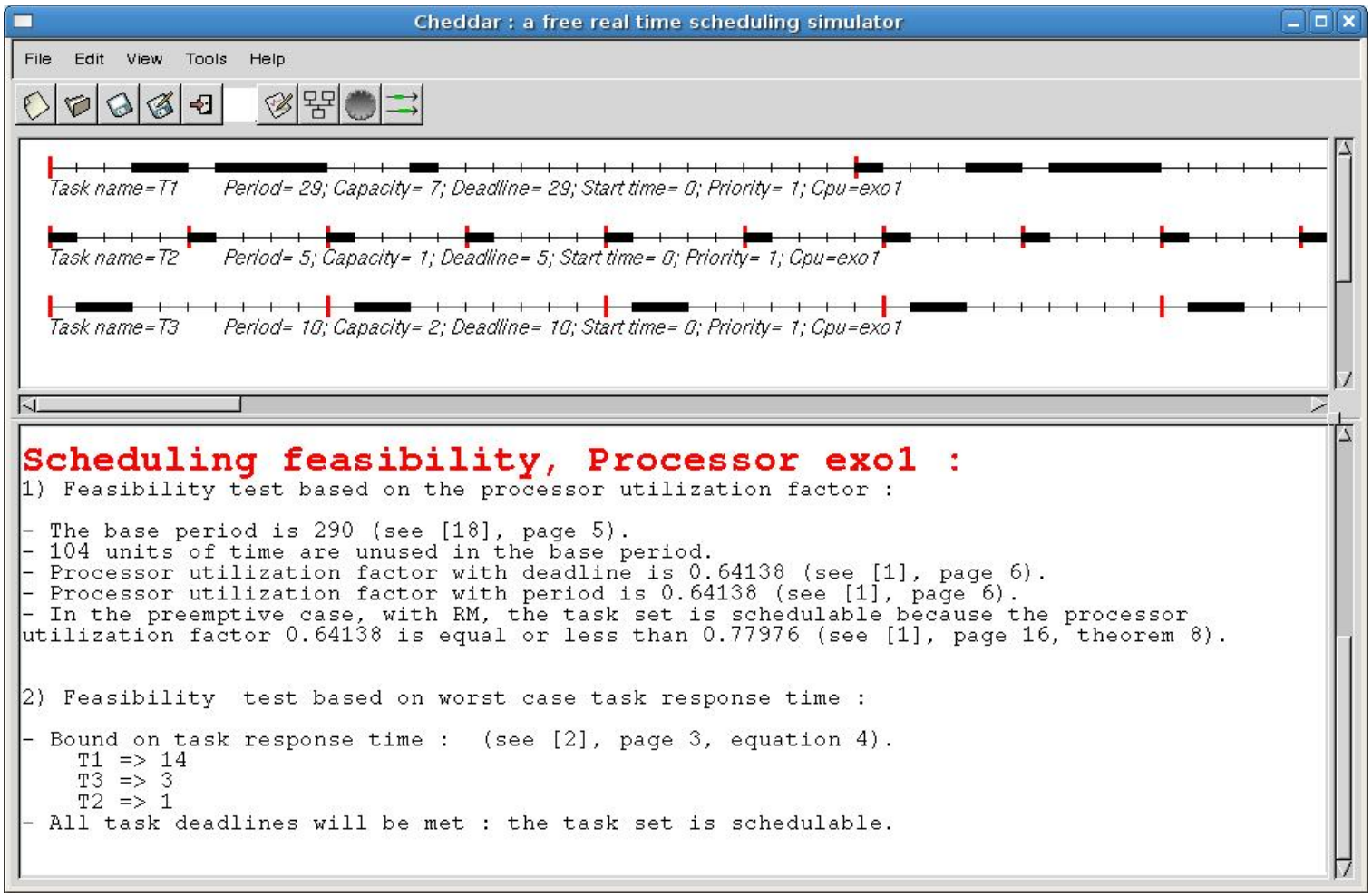


Алгоритмы планирования задач в ОС РВ

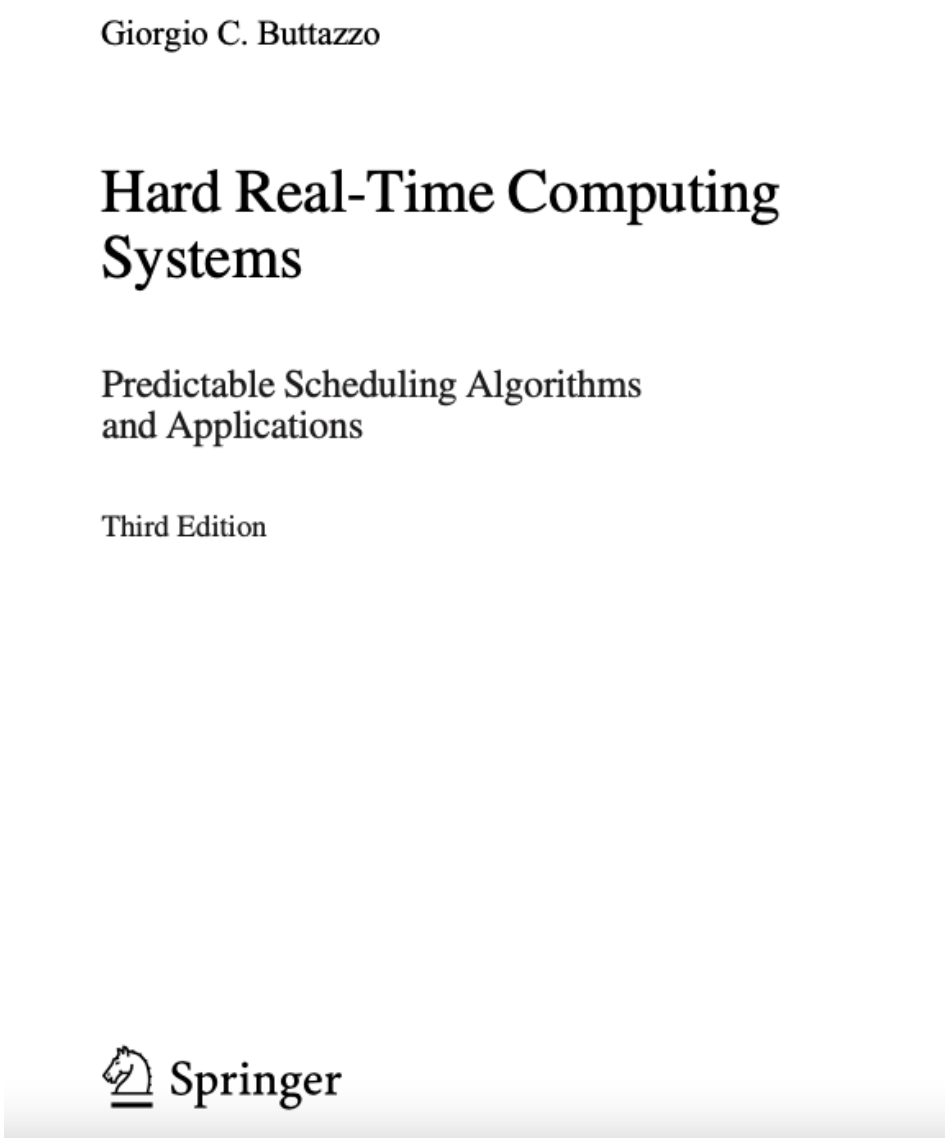
"Стандартизированные"алгоритмы

- RR = Round Robin = Планирование по кругу
- RMS = Rate-monotonic scheduling = Частотно-монотонное планирование
- EDF = Earliest deadline first scheduling = Планирование по ближайшему сроку завершения
- Least laxity first (LLF) = Наименее расслабленные сначала
- Least slack time (LST) = Наименьшее время простоя
- ... (для каждого набора задач может подходить свой алгоритм)

Анализ свойств планирования задач (Cheddar)



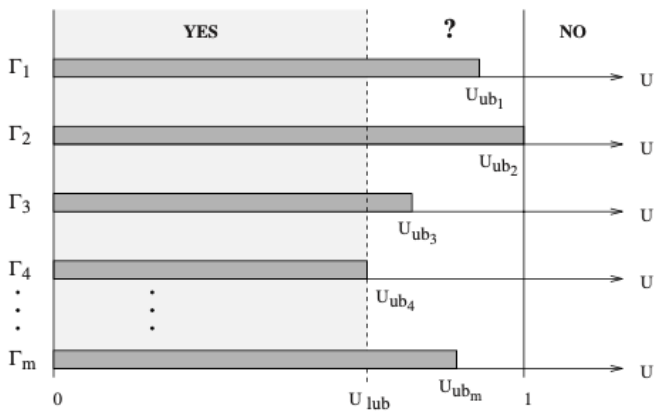
Анализ свойств планирования задач



Анализ свойств планирования задач (теория)

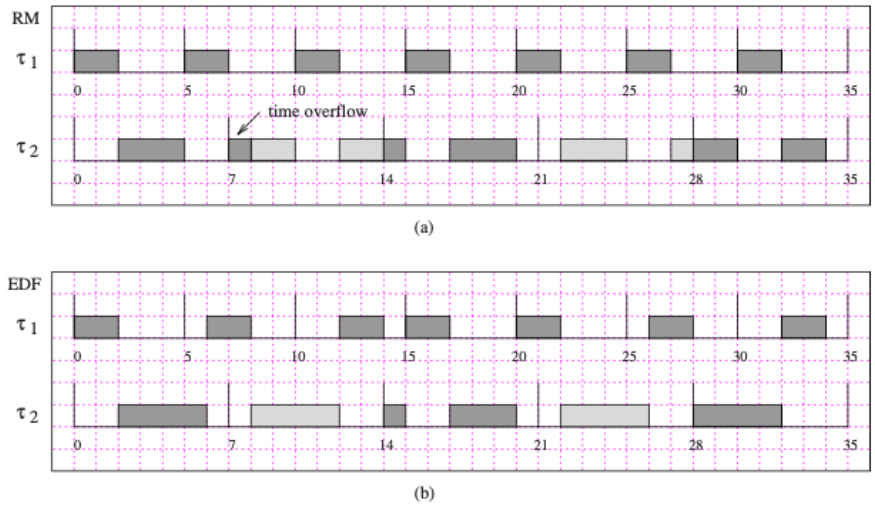
План доказательств того, что набор задач является планируемым конкретным алгоритмом:

- Расчет коэффициента использования процессора
- Расчет наименьшей верхней границы коэффициента использования процессора – минимум коэффициентов использования по всем наборам задач, которые полностью используют процессор



- Аналитический поиск: рассмотреть случаи, получить функцию, найти ее минимум

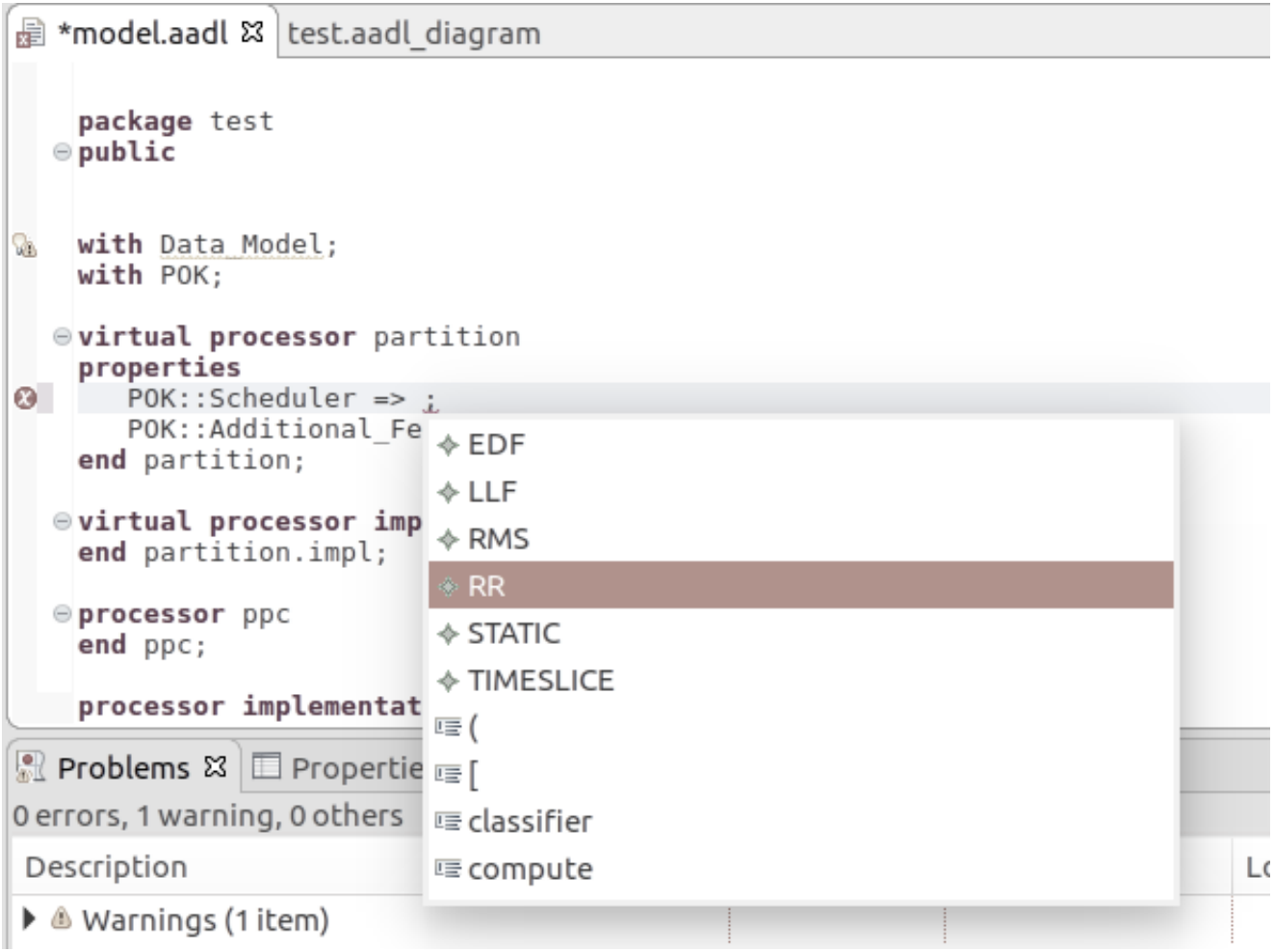
Анализ свойств планирования задач (пример из книги)



- $U = \frac{2}{5} + \frac{4}{7} = \frac{34}{35} = 0.97..$
- $U > 0.83$, планируемость набора задач с помощью алгоритма RMS не гарантирована;
- $U < 1$, планируемость набора задач с помощью алгоритма EDF гарантирована.

Партицированные (многораздельные) ОС

- Можно определить свой алгоритм планирования для каждого раздела



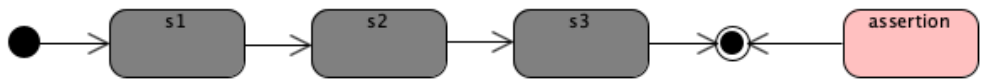
Партицированные (многораздельные) ОС

Моделирование и проверка поведения такой системы выполнено в виде выполнимой модели с использованием метода Model Checking, обсуждено на OS DAY 2020 и описано в статье "A Formal Model of a Partitioned Real-Time Operating System in Promela":

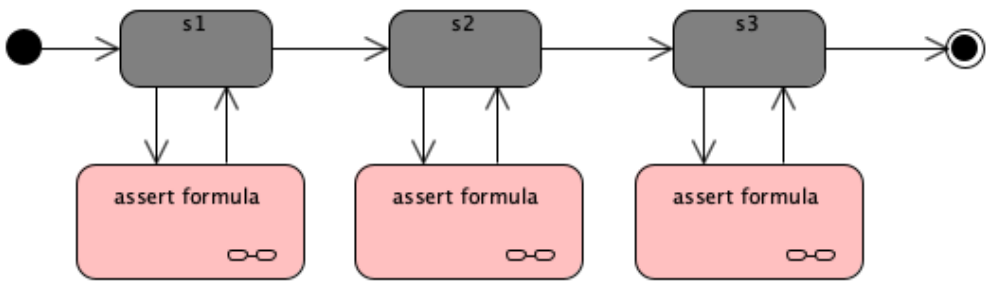
Модель включала программные прерывания для системных вызовов, работу с объектами типа семафор, два планировщика round robin с возможностью расширения стратегий планирования.

О Model Checking

- Метод проверки моделей получает на вход автоматную модель системы и автомат (полученный из темпоральной формулы) с требованиями
- Обычный процесс тестирования:



- Model Checking:



О Model Checking

- Системы вроде SPIN позволяют получить систему переходов из кода на DSL с ясной семантикой[4]
- Далее может быть проведен оптимизированный обход автомата (DFS) и проверка свойств над переменными системы
- Благодаря использованию ω-языков (автоматы Бюхи) можно проверять и бесконечно функционирующие системы
- Благодаря технологии Swarm Model Checking и ее потенциальным реализациям для CPU, GPU и FPGA можно проверять нарушение свойства в огромных системам с использованием хэшей состояний

[4]Natarajan, V., and Gerard J. Holzmann. "Outline for an operational semantics of Promela."The Spin Verification System 32 (1996): 133-152.

Доклад со SPIN 2019

Multifaceted Automated Analyses for Variability-Intensive Embedded Systems

Sami Lazreg
Visteon Electronics
Université Côte d'Azur
CNRS, I3S, France

Maxime Cordy*
SnT, University of Luxembourg
Luxembourg

Philippe Collet
Université Côte d'Azur
CNRS, I3S, France

Patrick Heymans
University of Namur
Belgium

Sébastien Mosser
Université Côte d'Azur
CNRS, I3S, France

***Abstract*—Embedded systems, like those found in the automotive domain, must comply with stringent functional and non-functional requirements. To fulfil these requirements, engineers are confronted with a plethora of design alternatives both at the software and hardware level, out of which they must select the optimal solution wrt. possibly-antagonistic quality attributes (e.g. cost of manufacturing vs. speed of execution). We propose a model-driven framework to assist engineers in this choice. It captures high-level specifications of the system in the form of variable dataflows and configurable hardware platforms. A mapping algorithm then derives the design space, i.e. the set of compatible pairs of application and platform variants, and a variability-aware executable model, which encodes the functional and non-functional behaviour of all viable system variants. Novel verification algorithms then pinpoint the optimal system variants efficiently. The benefits of our approach are evaluated through a**

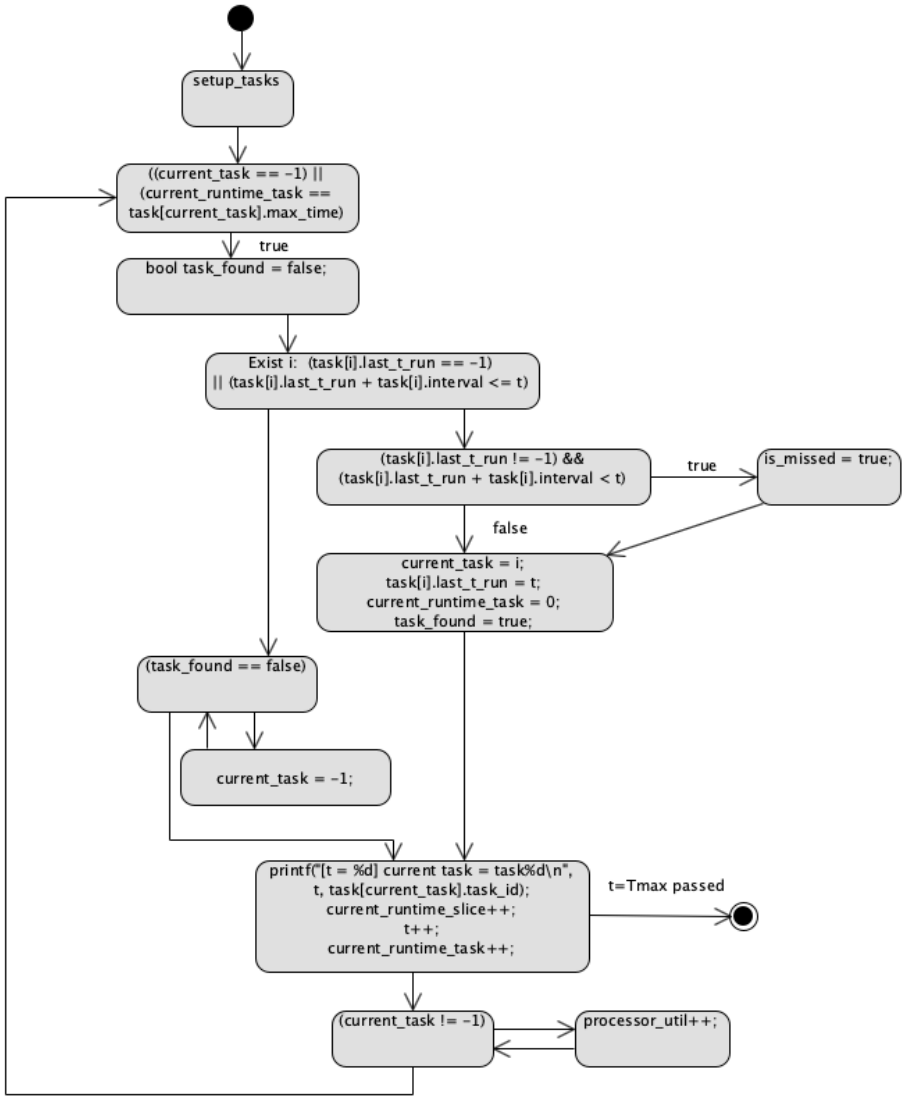
differing in, e.g., the size of the flowing data chunks, the ordering of the operation tasks, or the choice between alternative, functionally-equivalent tasks. Second, there exists a diversity of configurable hardware platforms which can differ, e.g., in memory capacities and processing pipelines. Third, there are various ways of mapping and deploying a given application on a specific platform, e.g., choose a processor to perform a given task or select a memory unit to store a given data.

This threefold variability is typical in automotive and many other kinds of embedded systems [1]. Unfortunately, it leads to a high number of variants (1,548,288 in this particular case), each of which represents a specific *system design alternative* (or *design* for short), that is, a specific mapping of a specific

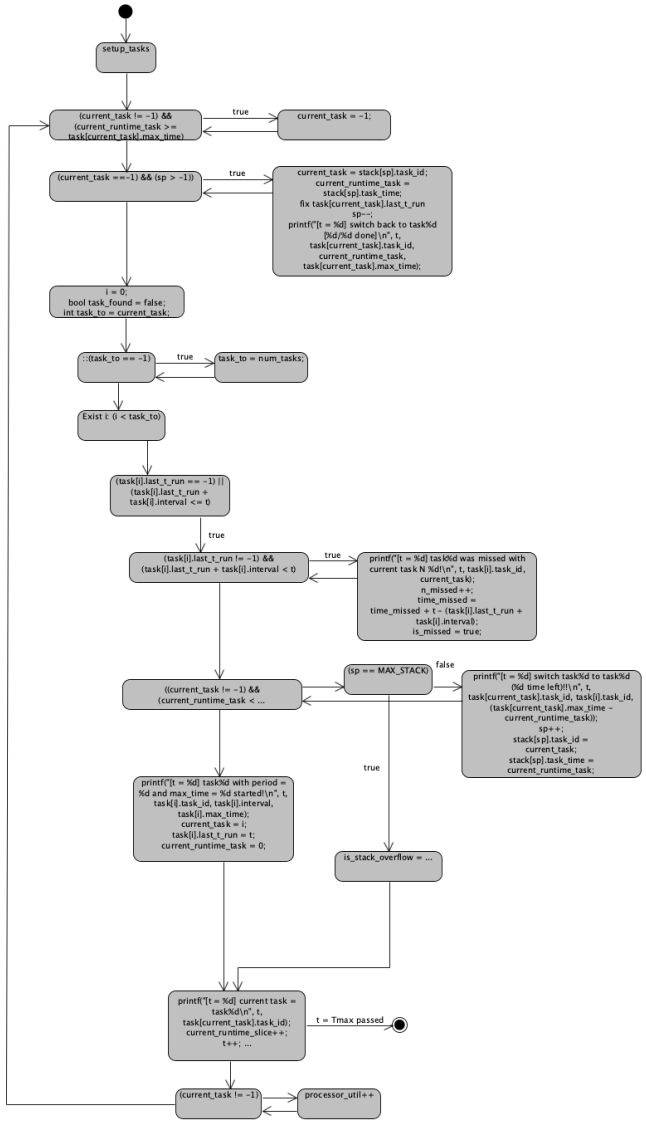
Решаемые задачи в нашей работе

- Реализация алгоритмов планирования набора задач на модельном языке верификатора
- Определение требований
- Эксперименты
- Определение перспектив

Реализация алгоритмов планирования – RMS (невытесняющий)



Реализация алгоритмов планирования – RMS (вытесняющий)



Реализация алгоритмов планирования

- Реализация алгоритмов планирования набора задач на модельном языке верификатора возможна
- Внутренний автомат SPIN меньше 200 состояний для двух видов стратегий вытеснения
- Далее возможно экспортировать систему переходов и в другие системы
- Поскольку мы явно имеем переменную “время”, то стратегия выбора задачи может быть написана в любом виде,лишь бы ее приняло средство верификации, если конечно не требуется показать близкую к эффективной реализацию с учетом этого параметра

К проверке свойств

В модель добавлены следующие переменные:

- Коэффициент использования процессора
- Признак и количество пропусков планирования
- Среднее количество пропущенных временных тактов

Соответственно, можно составлять формулы с данными переменными для автоматической проверки свойств.

Перспективы

- Реализация разных алгоритмов планирования и интеграция с прежде реализованной моделью
- Выбор оптимального алгоритма планирования для заданного набора задач
- Модели для усложненных алгоритмов планирования (доступ к общим ресурсам и т.д.)
- Переборный поиск наименьшей верхней границы использования процессора для набора задач

Оценка алгоритмов планирования набора задач в модели системы реального времени

Будьте здоровы!

serg_soft@mail.ru