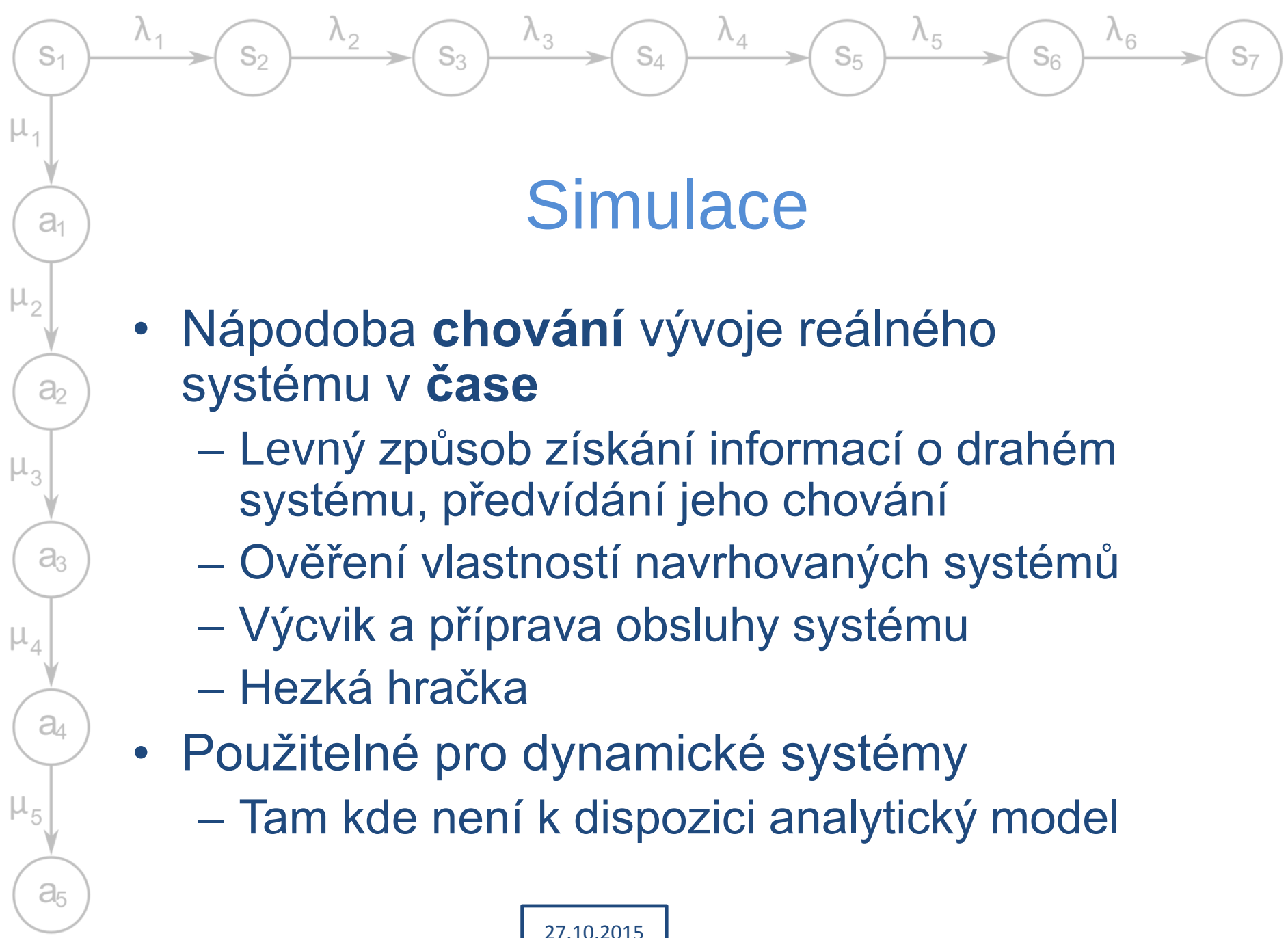
Simulace

Základy simulací

Simulace systémů hromadné obsluhy

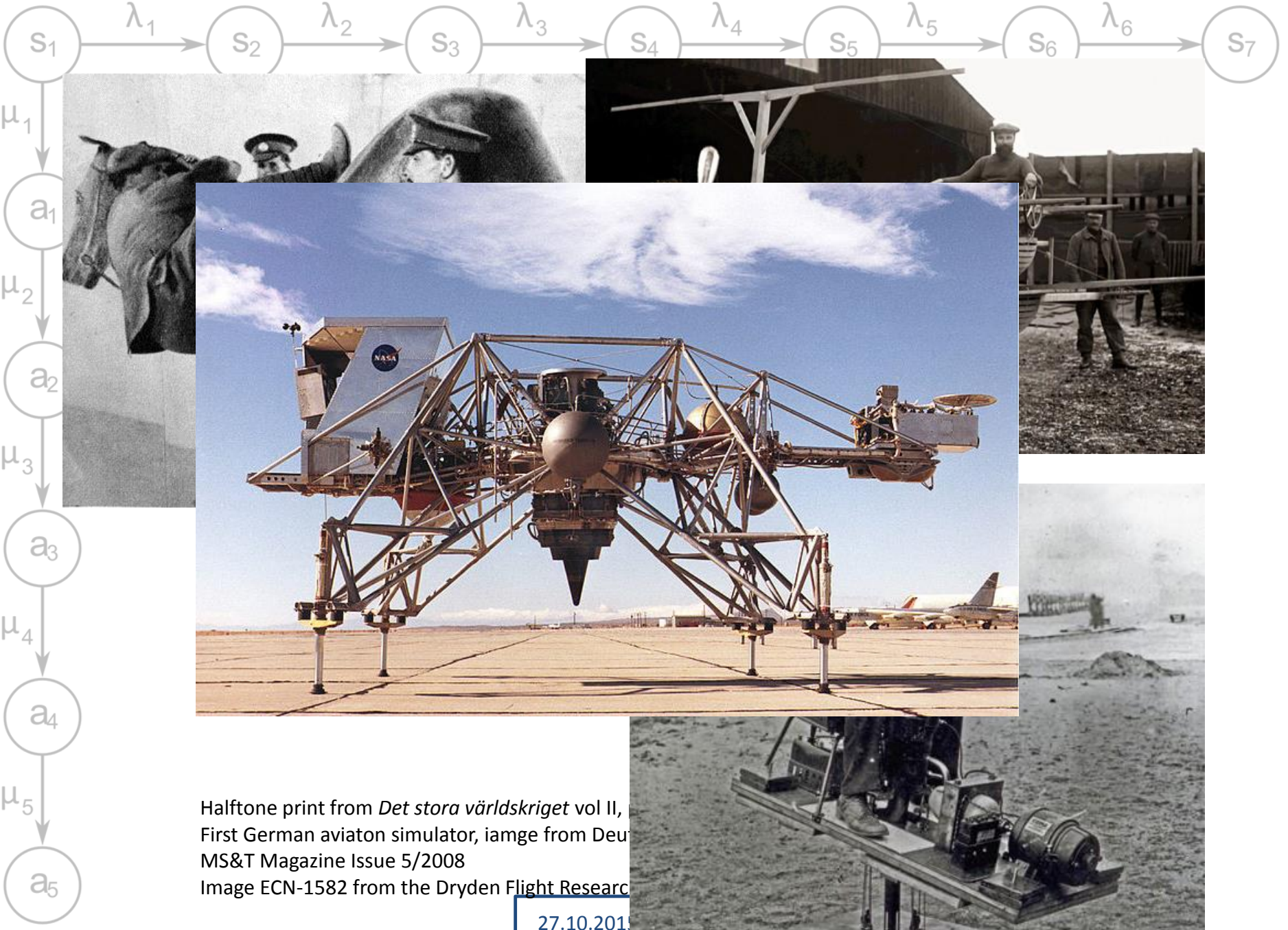
Richard Lipka

27.10. 2015

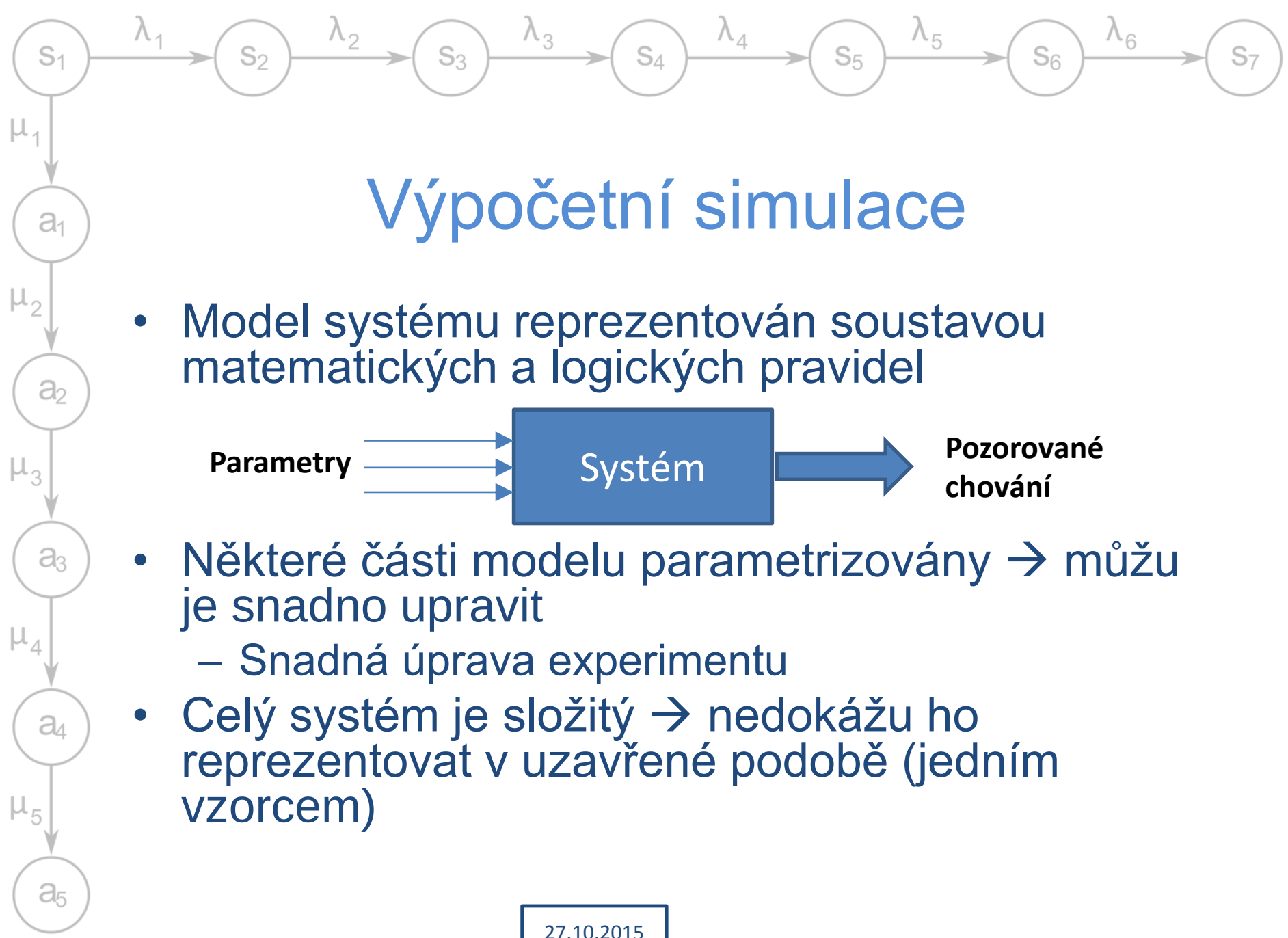


Simulace

- Nápodoba **chování** vývoje reálného systému v **čase**
  - Levný způsob získání informací o drahém systému, předvídání jeho chování
  - Ověření vlastností navrhovaných systémů
  - Výcvik a příprava obsluhy systému
  - Hezká hračka
- Použitelné pro dynamické systémy
  - Tam kde není k dispozici analytický model



Halftone print from *Det stora världskriget* vol II,
First German aviaton simulator, iamge from Deu
MS&T Magazine Issue 5/2008
Image ECN-1582 from the Dryden Flight Researc



Výpočetní simulace

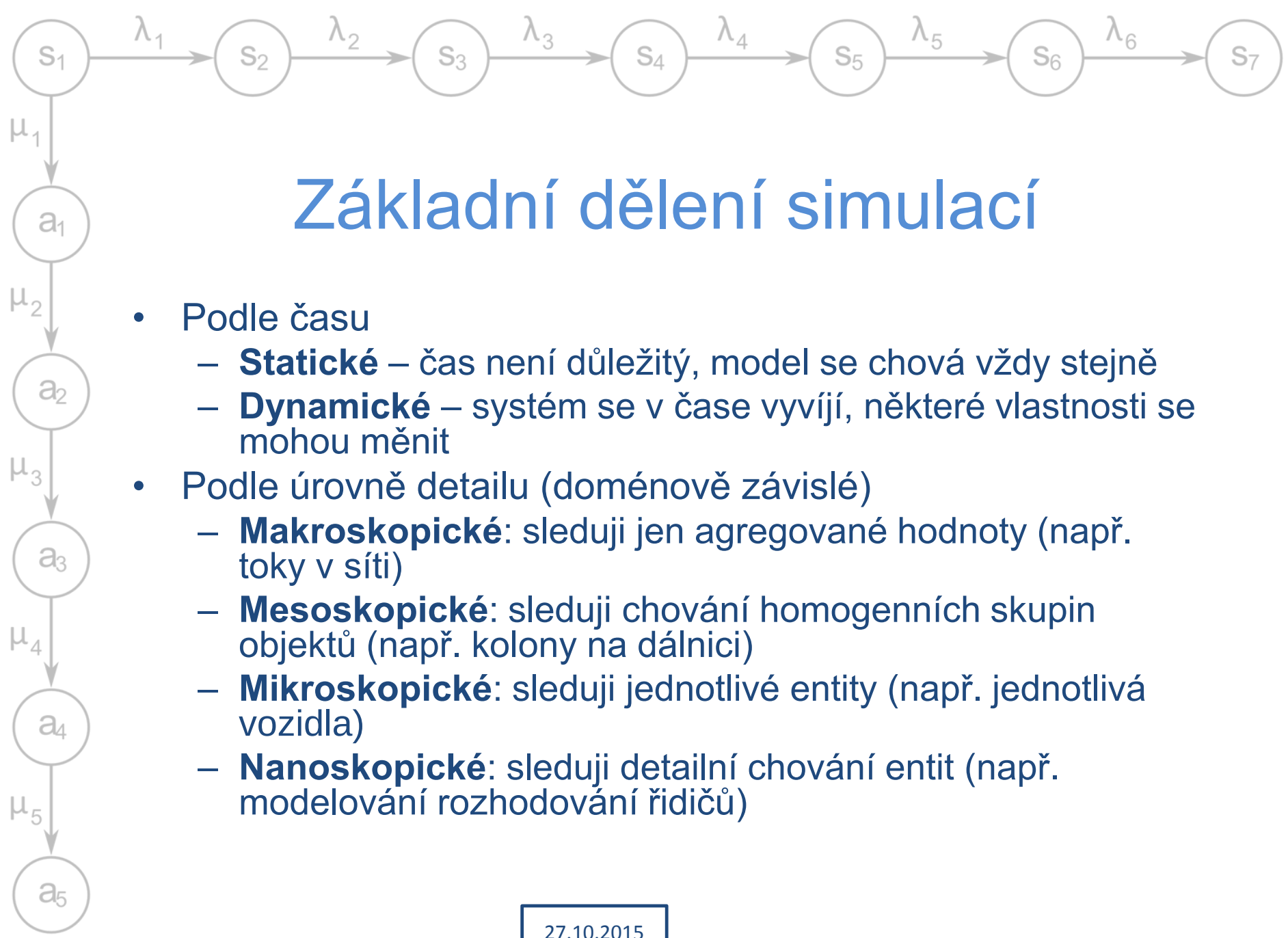
- Model systému reprezentován soustavou matematických a logických pravidel

- Některé části modelu parametrizovány → můžu je snadno upravit
  - Snadná úprava experimentu
- Celý systém je složitý → nedokážu ho reprezentovat v uzavřené podobě (jedním vzorcem)

Základ

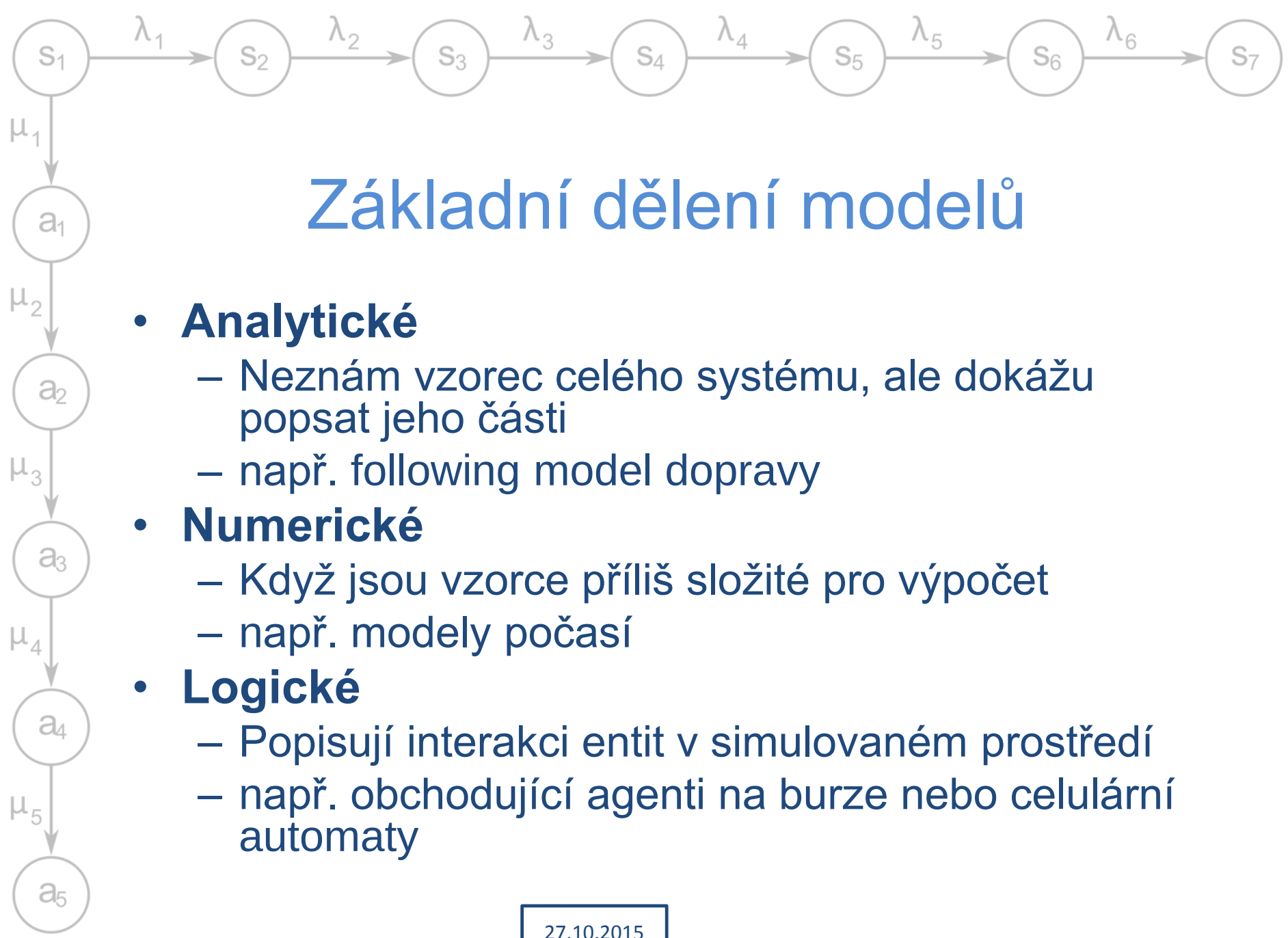
- Poč
  - **D**
    Ň
  - **S**
    N
- Poč
  - **S**
    o
  - **Diskrétní**: stav systému definován jen v některých bodech v čase

SongYhiPeng / http://www.orange.com/inspiration/next-please-by-zhipeng-song/
Oklahoma city community college, Biolabs



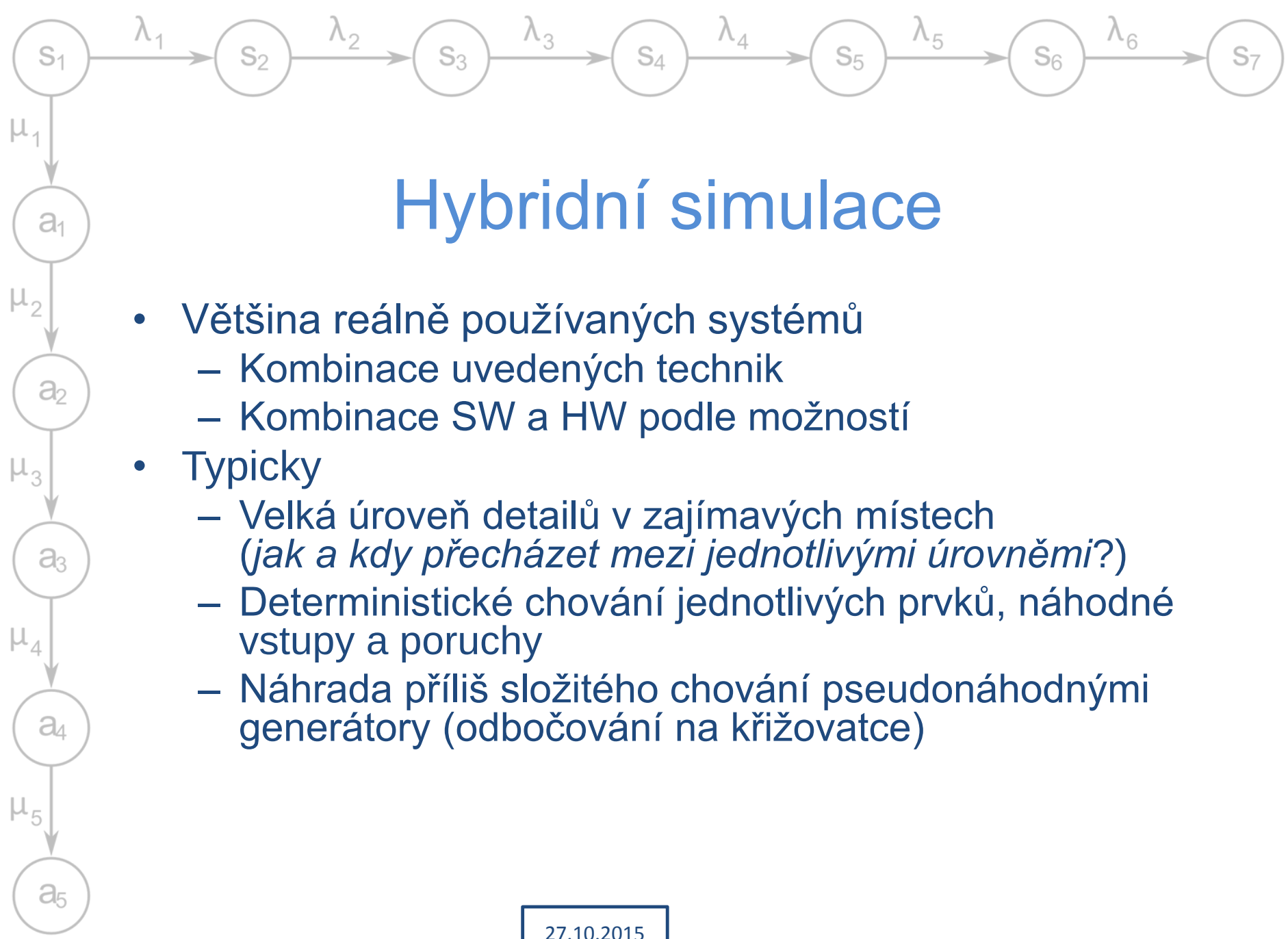
Základní dělení simulací

- Podle času
  - **Statické** – čas není důležitý, model se chová vždy stejně
  - **Dynamické** – systém se v čase vyvíjí, některé vlastnosti se mohou měnit
- Podle úrovně detailu (doménově závislé)
  - **Makroskopické**: sleduji jen agregované hodnoty (např. toky v síti)
  - **Mesoskopické**: sleduji chování homogenních skupin objektů (např. kolony na dálnici)
  - **Mikroskopické**: sleduji jednotlivé entity (např. jednotlivá vozidla)
  - **Nanoskopické**: sleduji detailní chování entit (např. modelování rozhodování řidičů)



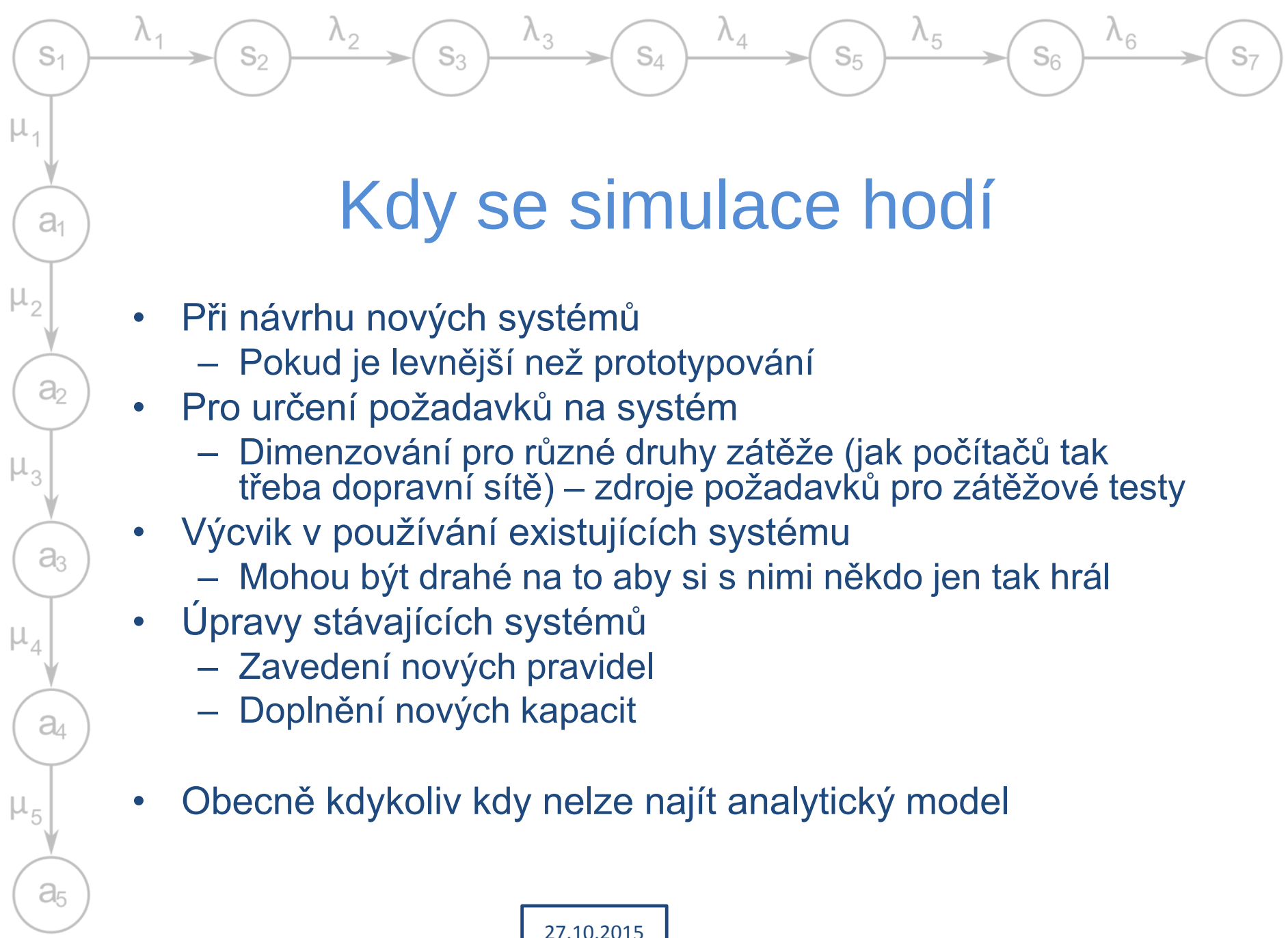
Základní dělení modelů

- **Analytické**
  - Neznám vzorec celého systému, ale dokážu popsat jeho části
  - např. following model dopravy
- **Numerické**
  - Když jsou vzorce příliš složité pro výpočet
  - např. modely počasí
- **Logické**
  - Popisují interakci entit v simulovaném prostředí
  - např. obchodující agenti na burze nebo celulární automaty



Hybridní simulace

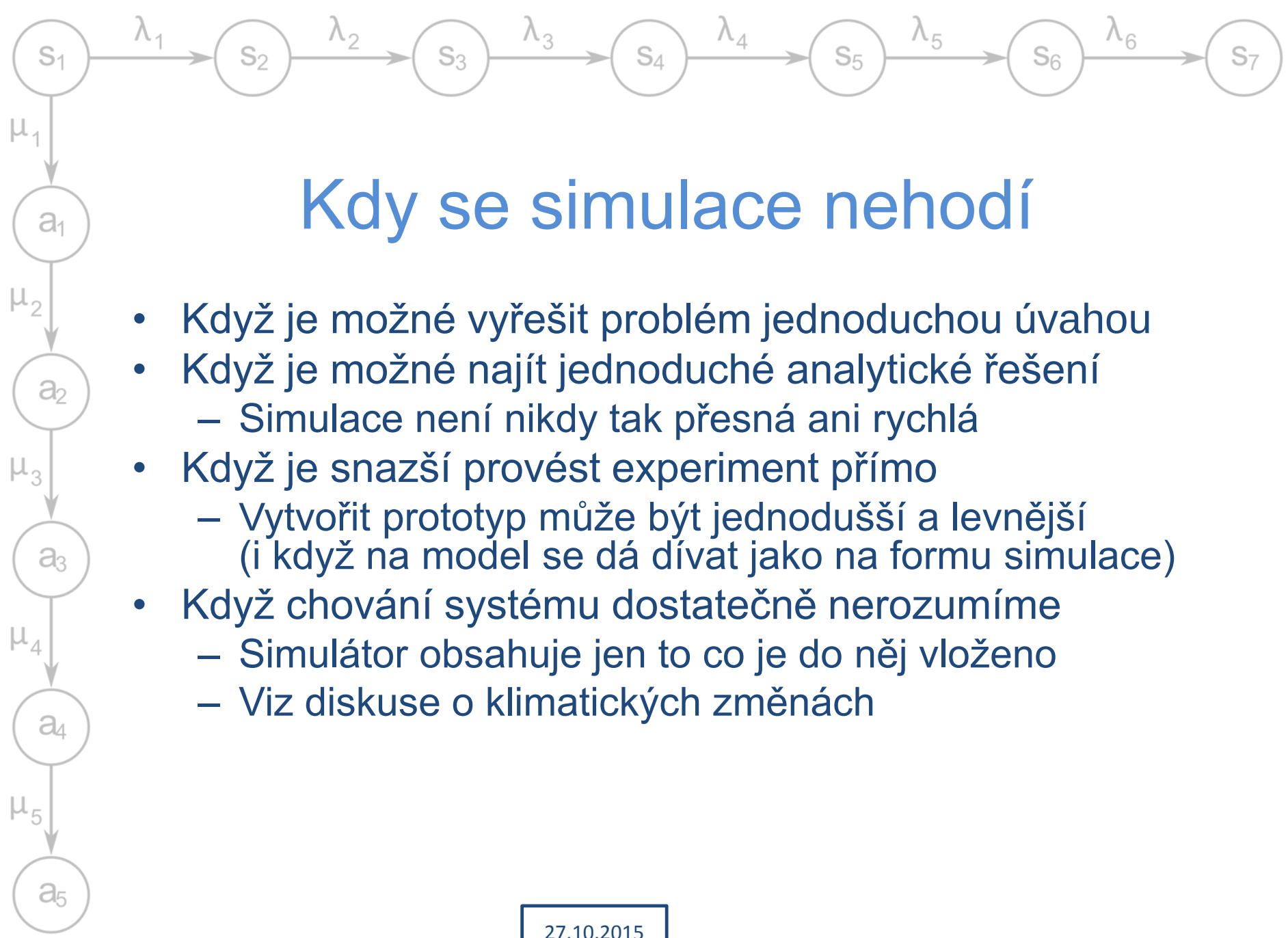
- Většina reálně používaných systémů
  - Kombinace uvedených technik
  - Kombinace SW a HW podle možností
- Typicky
  - Velká úroveň detailů v zajímavých místech (*jak a kdy přecházet mezi jednotlivými úrovněmi*?)
  - Deterministické chování jednotlivých prvků, náhodné vstupy a poruchy
  - Náhrada příliš složitého chování pseudonáhodnými generátory (odbočování na křižovatce)



Kdy se simulace hodí

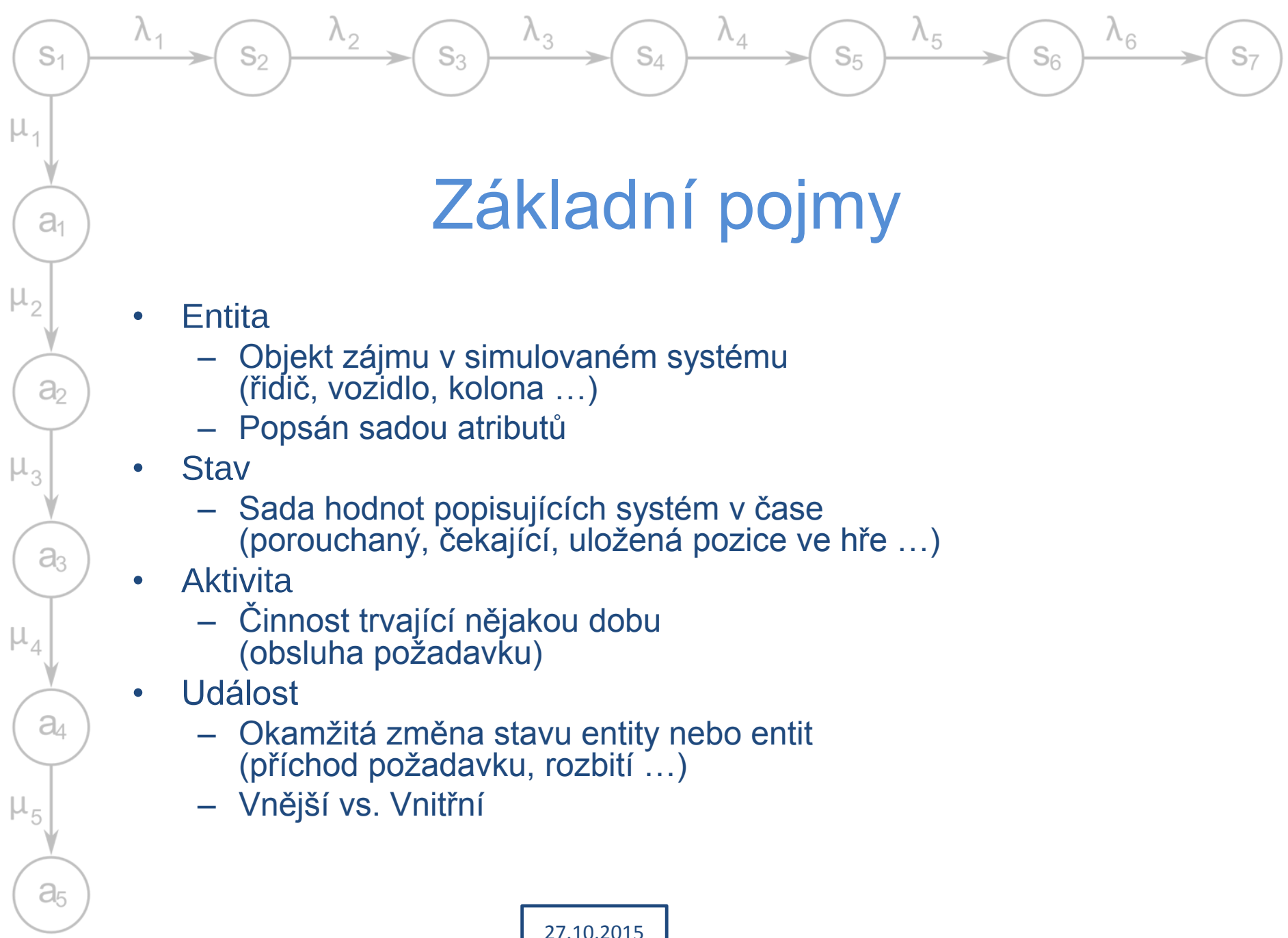
- Při návrhu nových systémů
  - Pokud je levnější než prototypování
- Pro určení požadavků na systém
  - Dimenzování pro různé druhy zátěže (jak počítačů tak třeba dopravní sítě) – zdroje požadavků pro zátěžové testy
- Výcvik v používání existujících systému
  - Mohou být drahé na to aby si s nimi někdo jen tak hrál
- Úpravy stávajících systémů
  - Zavedení nových pravidel
  - Doplnění nových kapacit

- Obecně kdykoliv kdy nelze najít analytický model



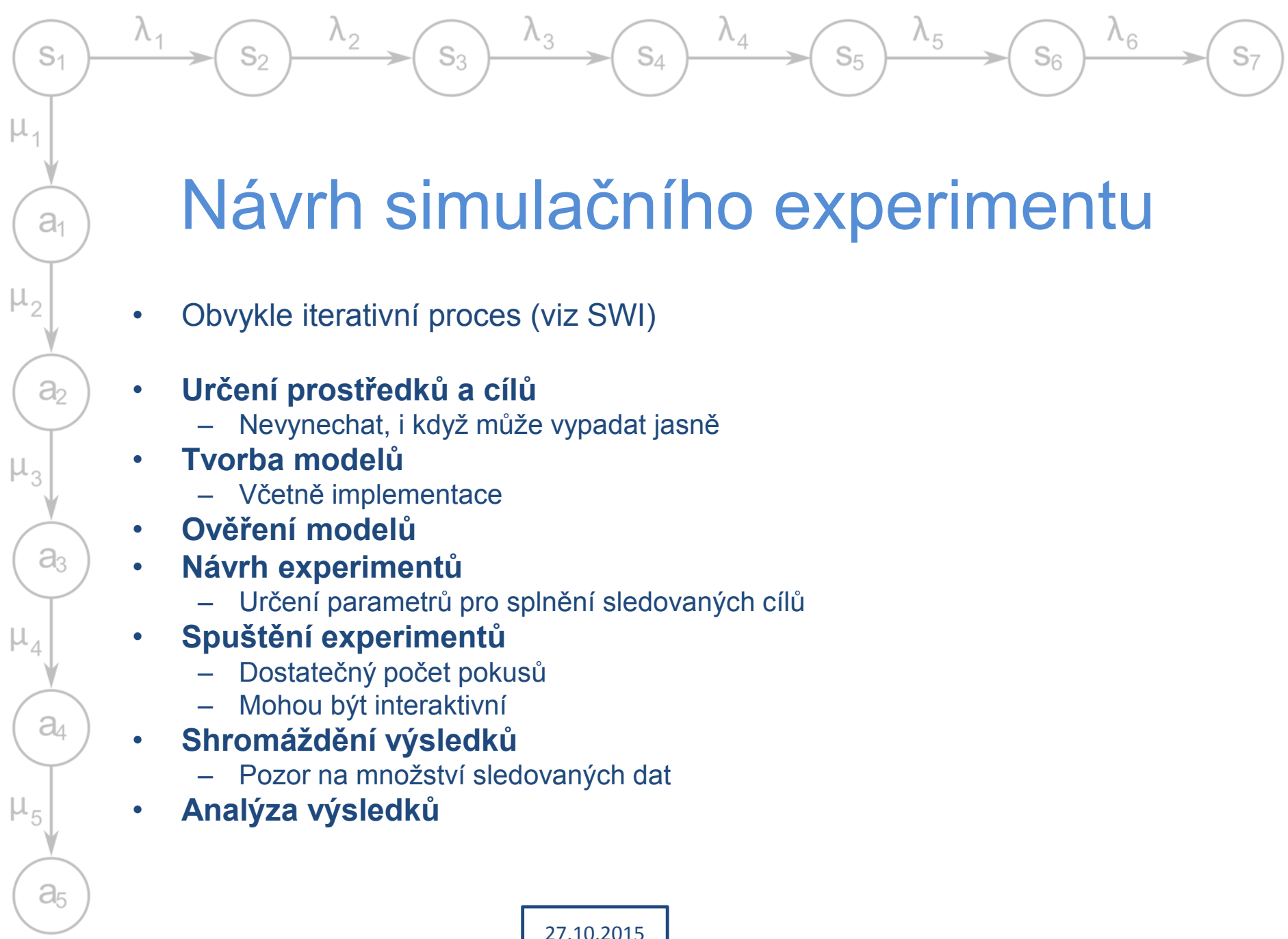
Kdy se simulace nehodí

- Když je možné vyřešit problém jednoduchou úvahou
- Když je možné najít jednoduché analytické řešení
  - Simulace není nikdy tak přesná ani rychlá
- Když je snazší provést experiment přímo
  - Vytvořit prototyp může být jednodušší a levnější (i když na model se dá dívat jako na formu simulace)
- Když chování systému dostatečně nerozumíme
  - Simulátor obsahuje jen to co je do něj vloženo
  - Viz diskuse o klimatických změnách



Základní pojmy

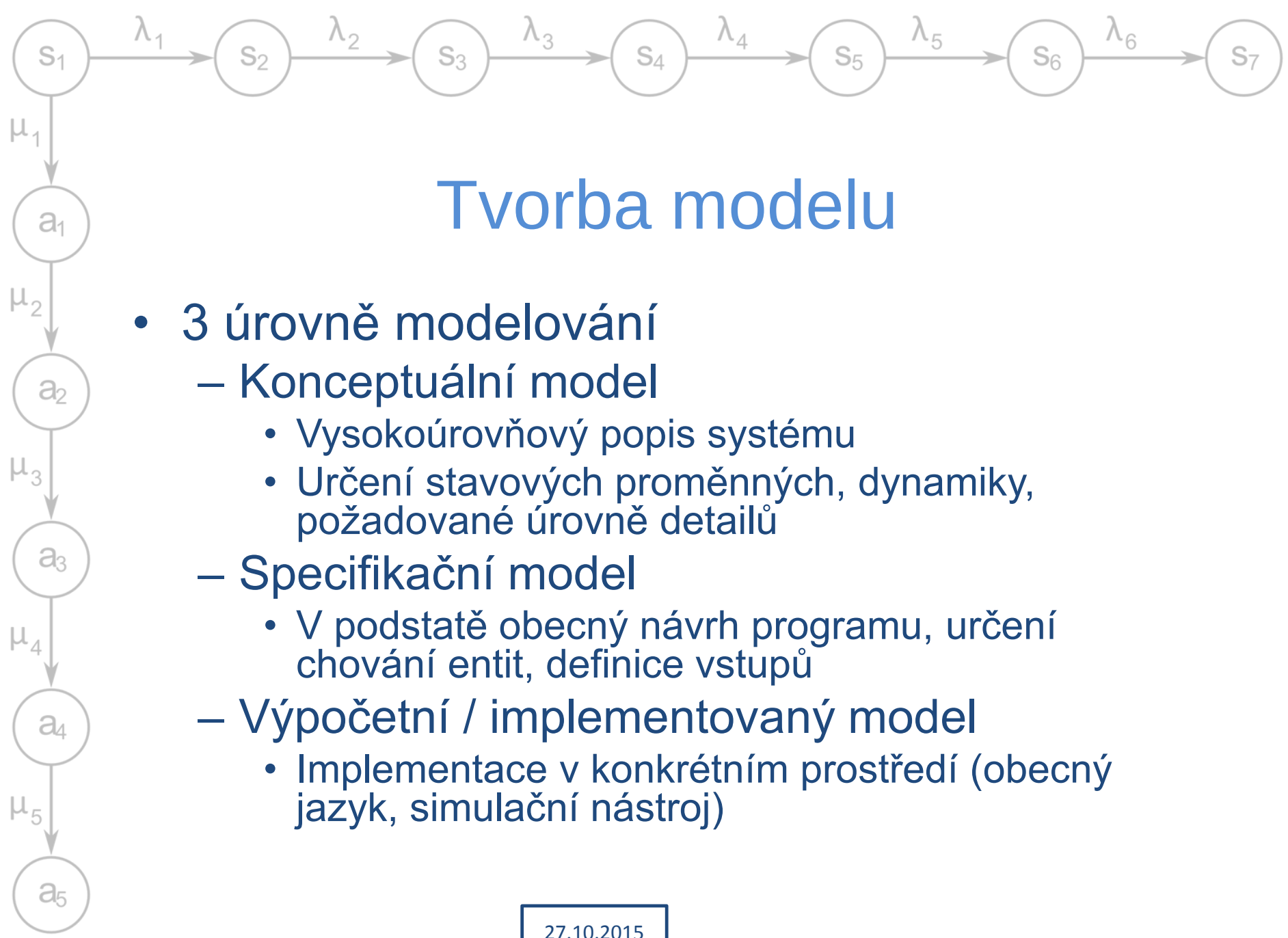
- Entita
  - Objekt zájmu v simulovaném systému (řidič, vozidlo, kolona …)
  - Popsán sadou atributů
- Stav
  - Sada hodnot popisujících systém v čase (porouchaný, čekající, uložená pozice ve hře …)
- Aktivita
  - Činnost trvající nějakou dobu (obsluha požadavku)
- Událost
  - Okamžitá změna stavu entity nebo entit (příchod požadavku, rozbití …)
  - Vnější vs. Vnitřní



Návrh simulačního experimentu

- Obvykle iterativní proces (viz SWI)

- **Určení prostředků a cílů**
  - Nevynechat, i když může vypadat jasně
- **Tvorba modelů**
  - Včetně implementace
- **Ověření modelů**
- **Návrh experimentů**
  - Určení parametrů pro splnění sledovaných cílů
- **Spuštění experimentů**
  - Dostatečný počet pokusů
  - Mohou být interaktivní
- **Shromáždění výsledků**
  - Pozor na množství sledovaných dat
- **Analýza výsledků**

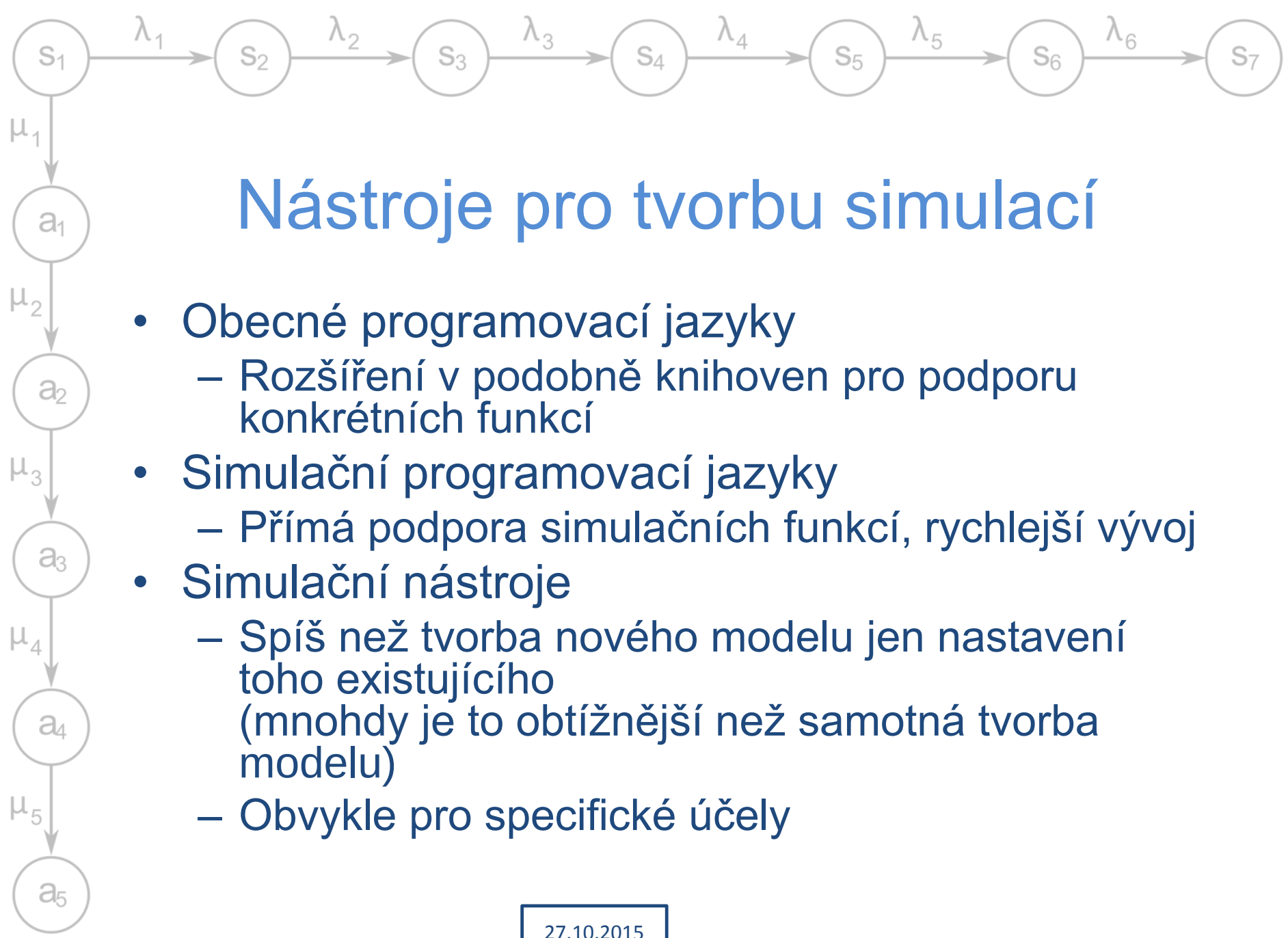


Tvorba modelu

- 3 úrovně modelování
  - Konceptuální model
    - Vysokoúrovňový popis systému
    - Určení stavových proměnných, dynamiky, požadované úrovně detailů
  - Specifikační model
    - V podstatě obecný návrh programu, určení chování entit, definice vstupů
  - Výpočetní / implementovaný model
    - Implementace v konkrétním prostředí (obecný jazyk, simulační nástroj)

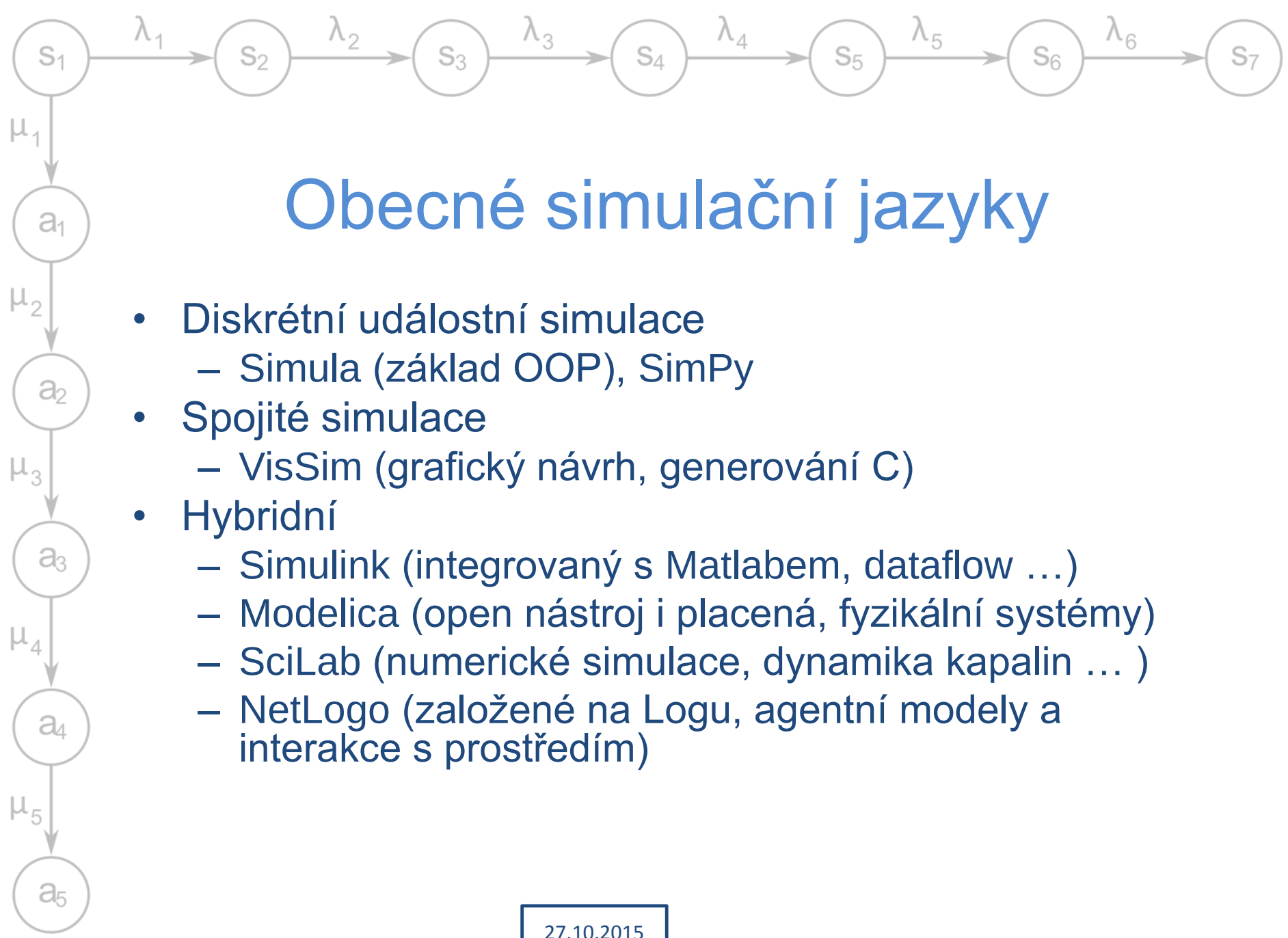
Ověření modelů

- **Verifikace**
  - Je implementovaný model konzistentní se specifikačním? (implementoval jsem model správně?)
  - Standardní SWI techniky
  - Nalezená chyba = bug, „snadná oprava“
- **Validace**
  - Je implementovaný model konzistentní se zkoumaným systémem (implementoval jsem správný model?)
  - Dokáže expert odlišit reálná měření od výstupů z modelu?
  - Nalezená chyba = model v něčem neodpovídá modelované skutečnosti, vadí mi to?



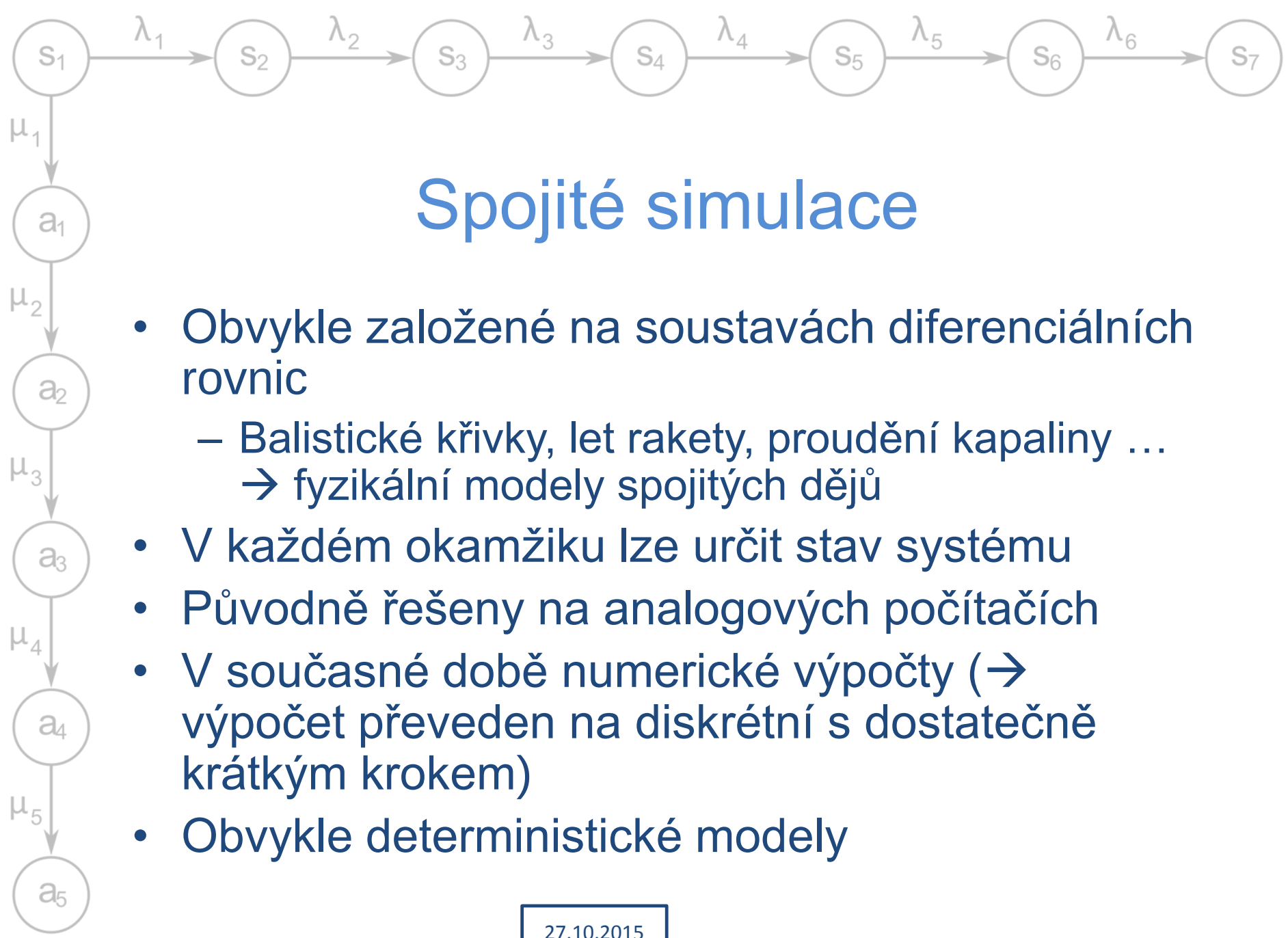
Nástroje pro tvorbu simulací

- Obecné programovací jazyky
  - Rozšíření v podobně knihoven pro podporu konkrétních funkcí
- Simulační programovací jazyky
  - Přímá podpora simulačních funkcí, rychlejší vývoj
- Simulační nástroje
  - Spíš než tvorba nového modelu jen nastavení toho existujícího
    (mnohdy je to obtížnější než samotná tvorba modelu)
  - Obvykle pro specifické účely



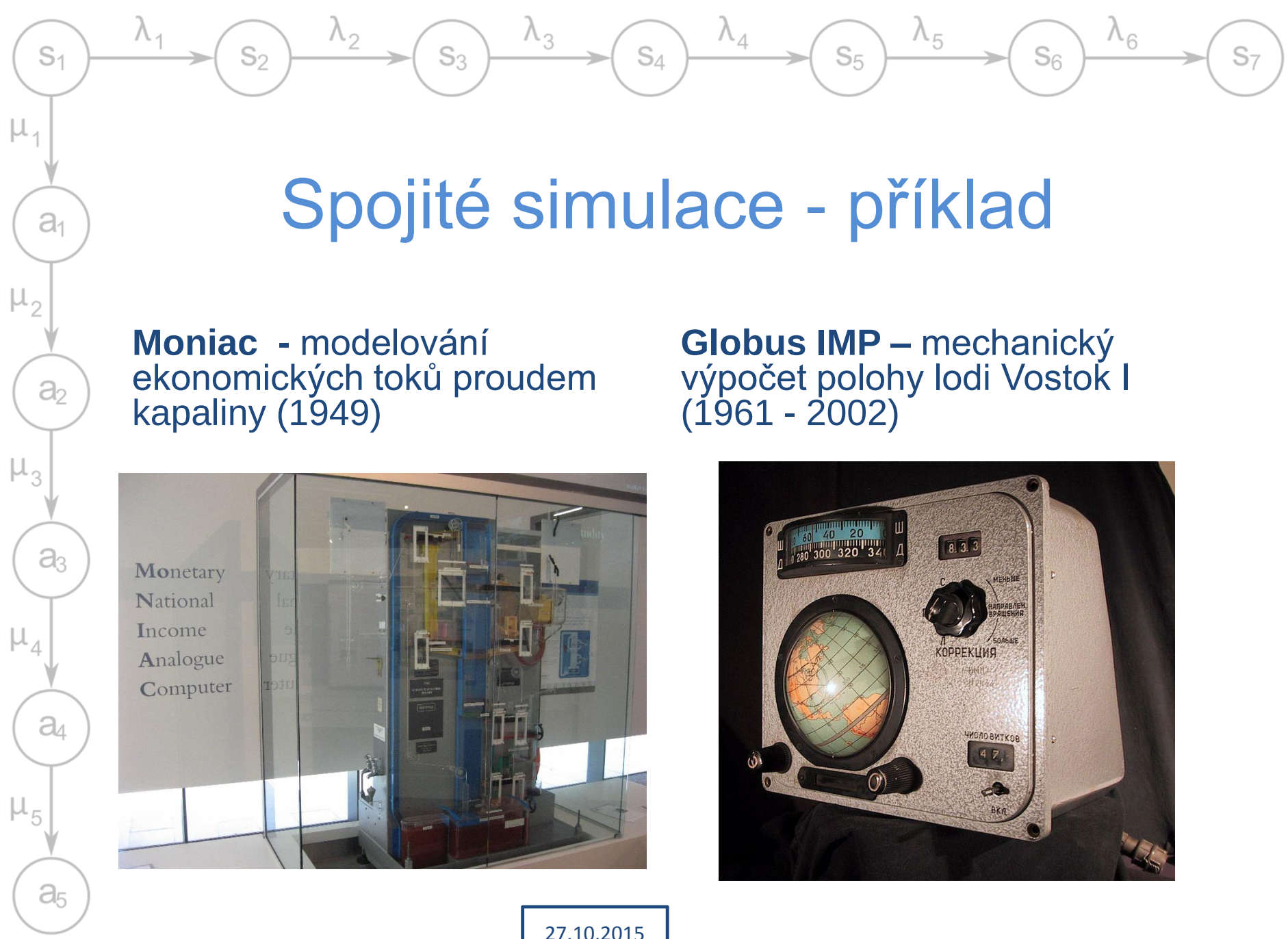
Obecné simulační jazyky

- Diskrétní událostní simulace
  - Simula (základ OOP), SimPy
- Spojité simulace
  - VisSim (grafický návrh, generování C)
- Hybridní
  - Simulink (integrovaný s Matlabem, dataflow …)
  - Modelica (open nástroj i placená, fyzikální systémy)
  - SciLab (numerické simulace, dynamika kapalin …)
  - NetLogo (založené na Logu, agentní modely a interakce s prostředím)



Spojité simulace

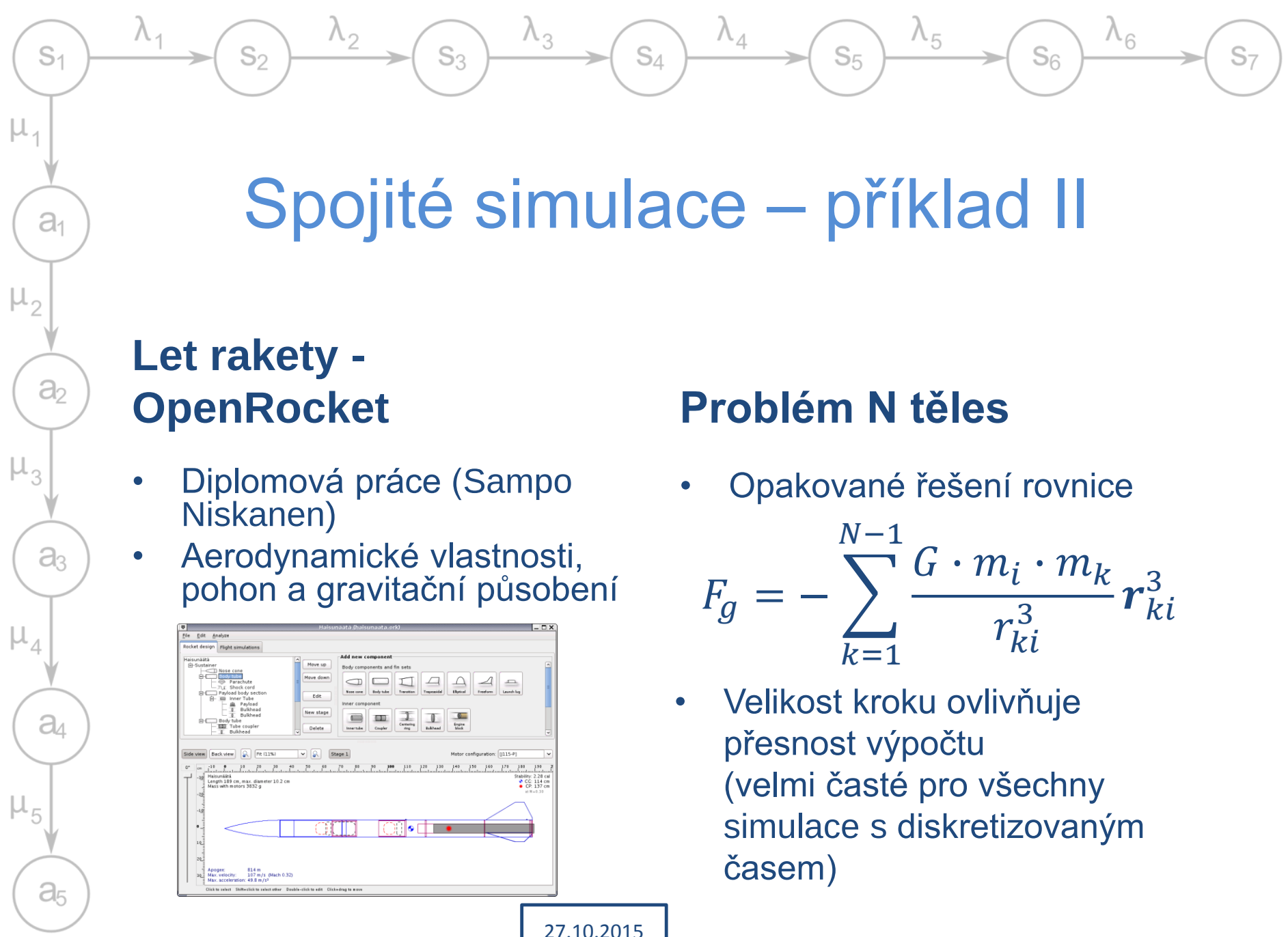
- Obvykle založené na soustavách diferenciálních rovnic
  - Balistické křivky, let rakety, proudění kapaliny … → fyzikální modely spojitých dějů
- V každém okamžiku lze určit stav systému
- Původně řešeny na analogových počítačích
- V současné době numerické výpočty (→ výpočet převeden na diskrétní s dostatečně krátkým krokem)
- Obvykle deterministické modely



Spojité simulace - příklad

Moniac - modelování ekonomických toků proudem kapaliny (1949)

Globus IMP – mechanický výpočет polohy lodi Vostok I (1961 - 2002)



Spojité simulace – příklad II

Let rakety - OpenRocket

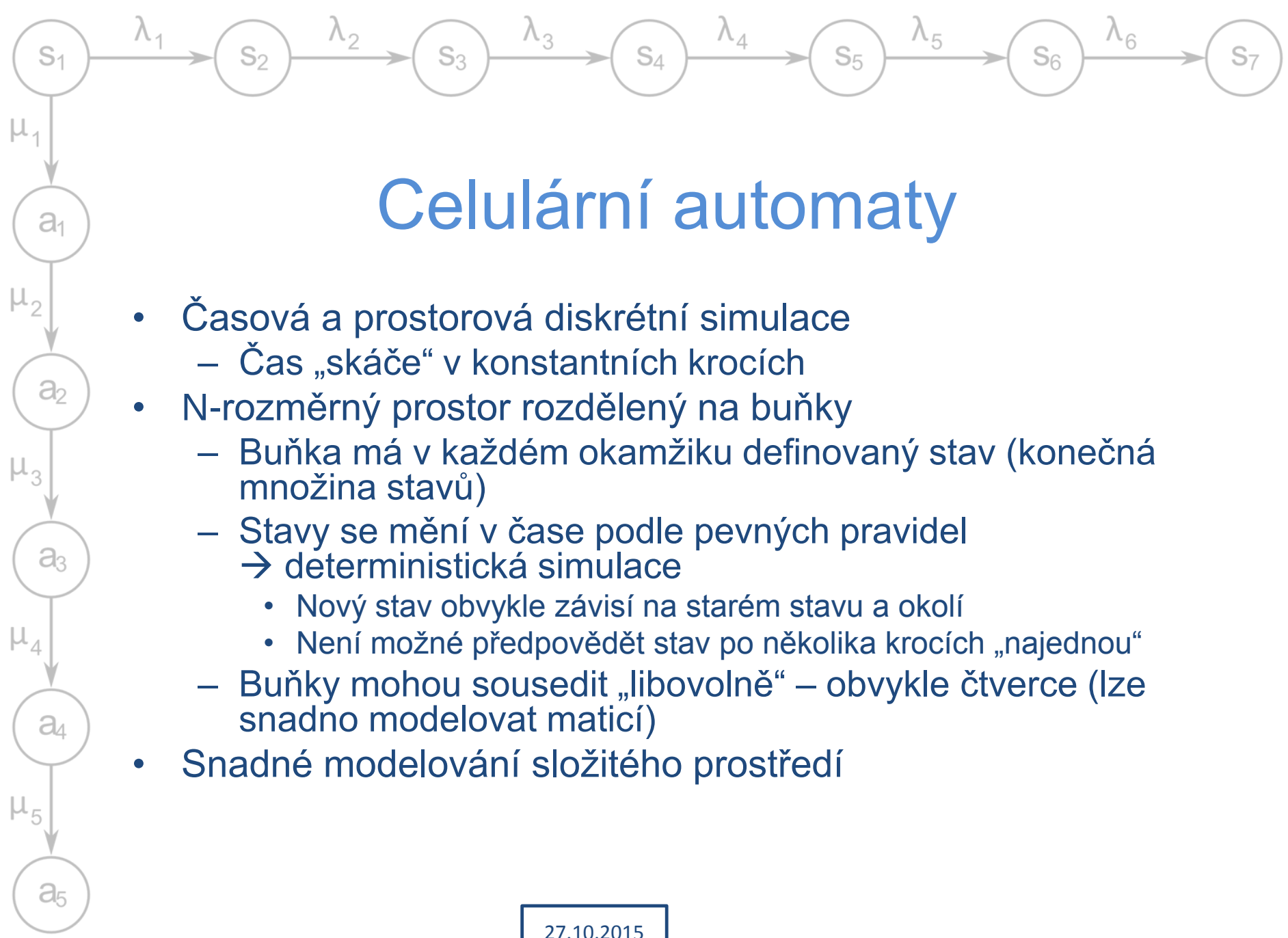
- Diplomová práce (Sampo Niskanen)
- Aerodynamické vlastnosti, pohon a gravitační působení

Problém N těles

- Opakované řešení rovnice

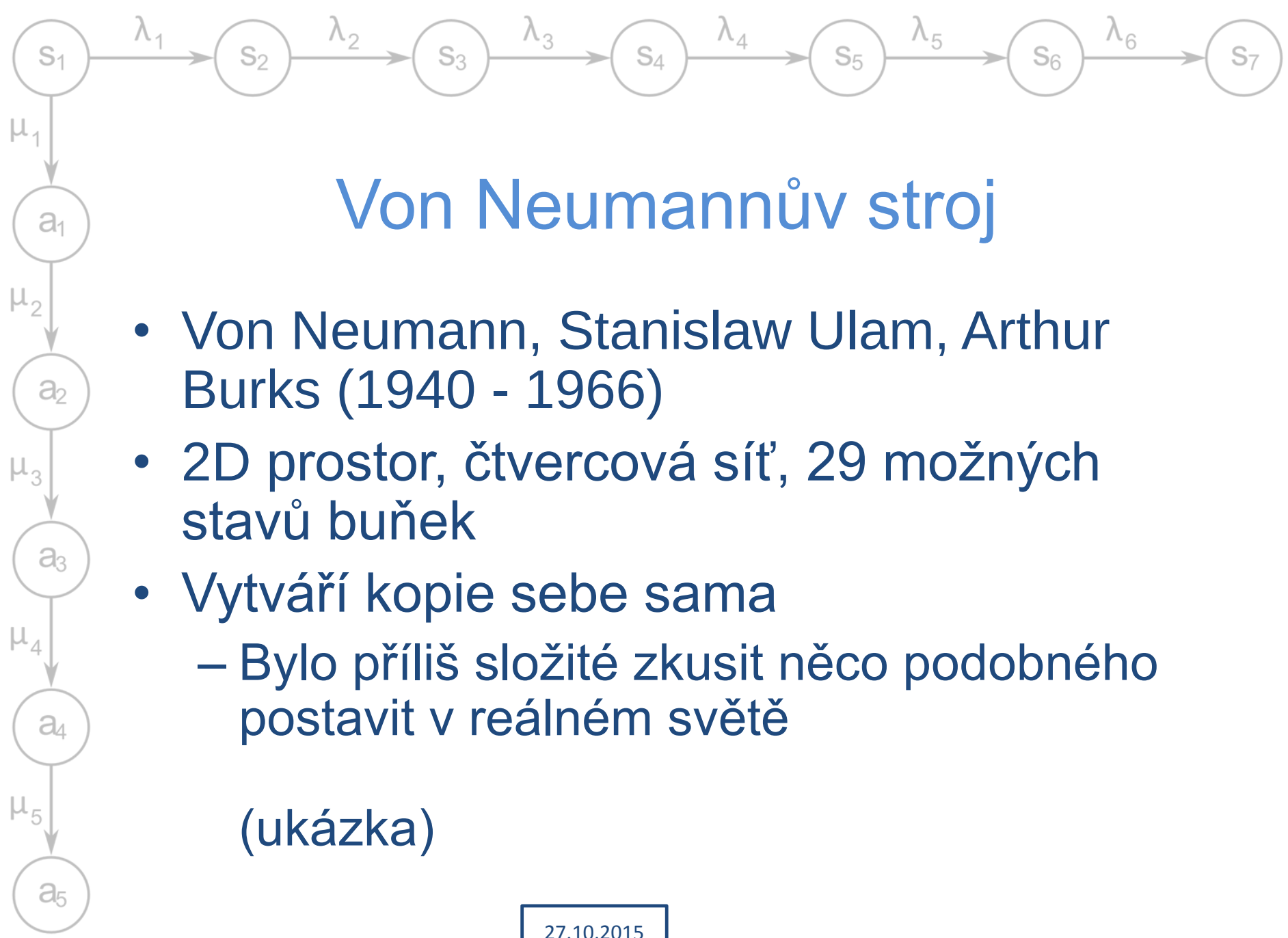
$$F_g = -\sum_{k=1}^{N-1} \frac{G \cdot m_i \cdot m_k}{r_{ki}^3} \boldsymbol{r}_{ki}^3$$

- Velikost kroku ovlivňuje přesnost výpočtu (velmi časté pro všechny simulace s diskretizovaným časem)



Celulární automaty

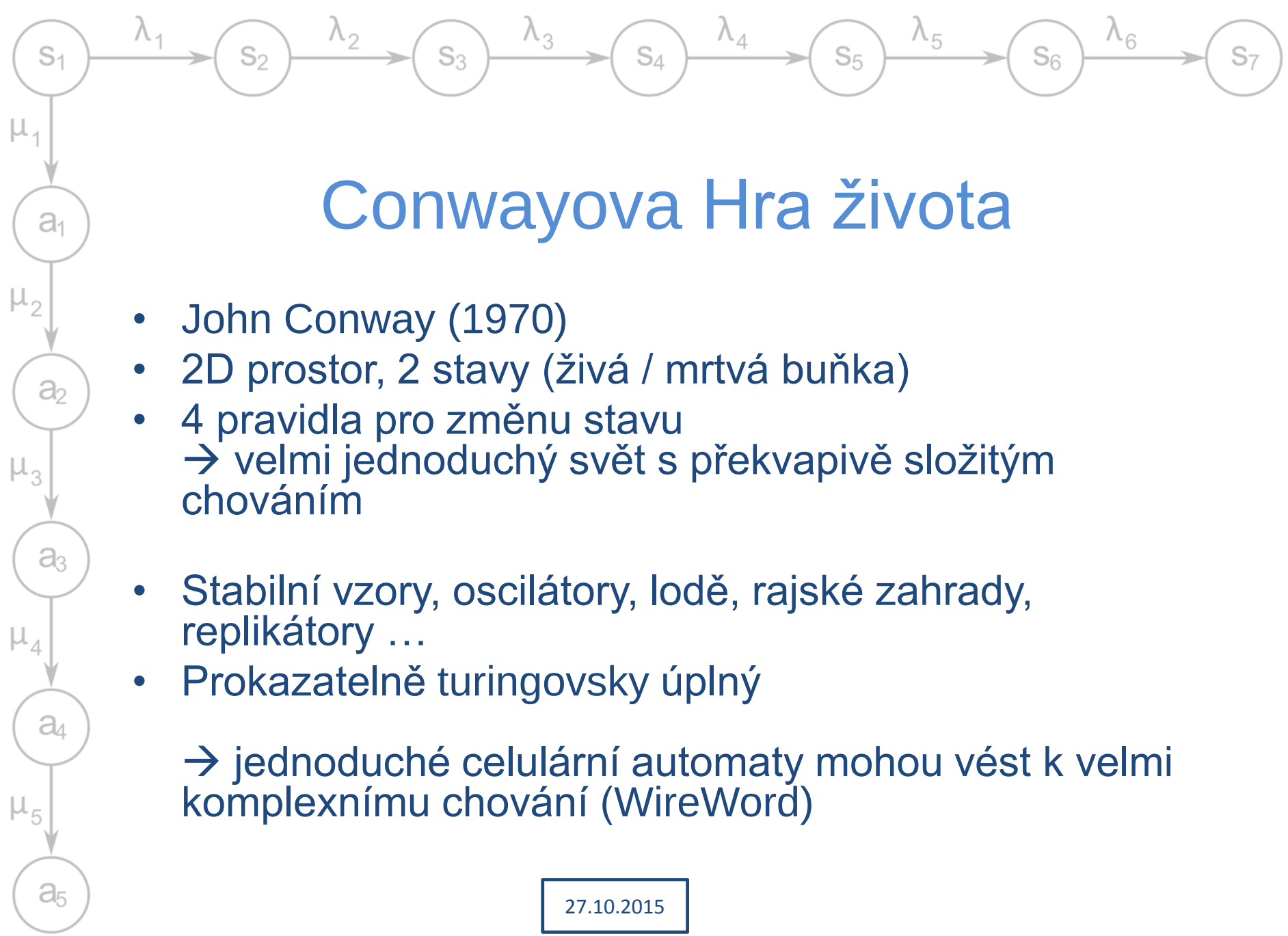
- Časová a prostorová diskrétní simulace
  - Čas „skáče“ v konstantních krocích
- N-rozměrný prostor rozdělený na buňky
  - Buňka má v každém okamžiku definovaný stav (konečná množina stavů)
  - Stavy se mění v čase podle pevných pravidel → deterministická simulace
    - Nový stav obvykle závisí na starém stavu a okolí
    - Není možné předpovědět stav po několika krocích „najednou“
  - Buňky mohou sousedit „libovolně“ – obvykle čtverce (lze snadno modelovat maticí)
- Snadné modelování složitého prostředí



Von Neumannův stroj

- Von Neumann, Stanislaw Ulam, Arthur Burks (1940 - 1966)
- 2D prostor, čtvercová síť, 29 možných stavů buněk
- Vytváří kopie sebe sama
  - Bylo příliš složité zkusit něco podobného postavit v reálném světě

  (ukázka)

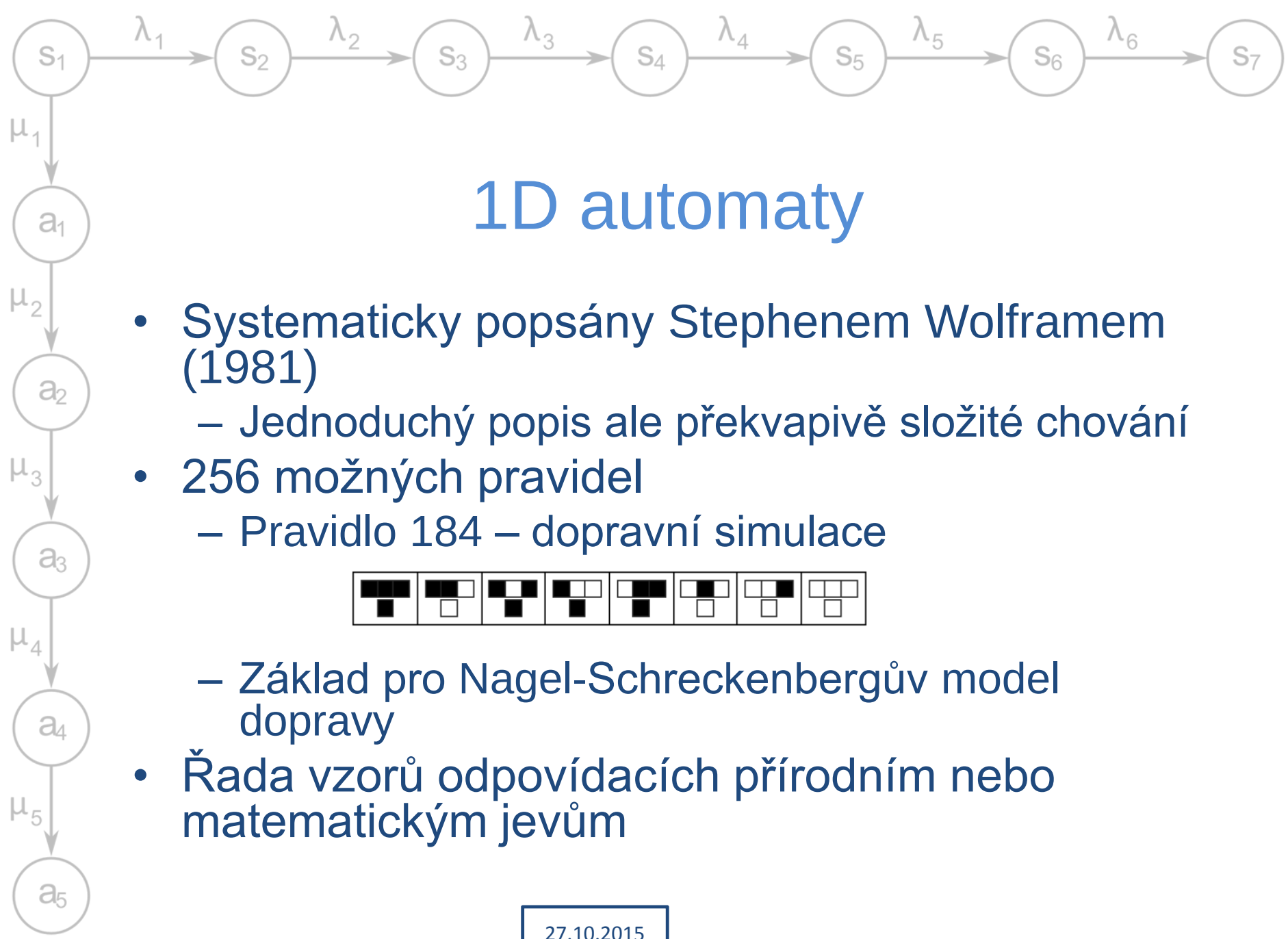


Conwayova Hra života

- John Conway (1970)
- 2D prostor, 2 stavy (živá / mrtvá buňka)
- 4 pravidla pro změnu stavu
  → velmi jednoduchý svět s překvapivě složitým chováním

- Stabilní vzory, oscilátory, lodě, rajské zahrady, replikátory …
- Prokazatelně turingovsky úplný

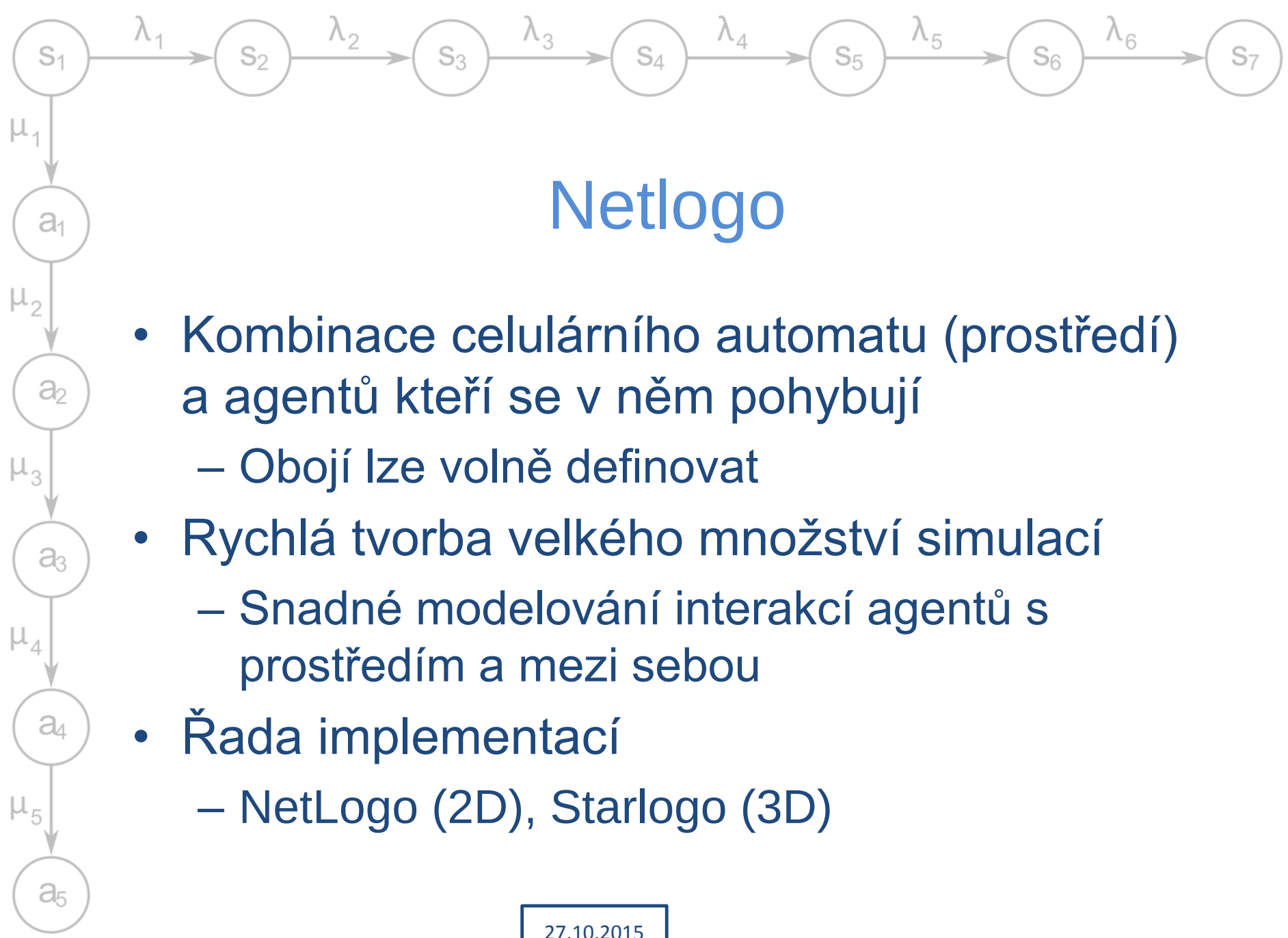
→ jednoduché celulární automaty mohou vést k velmi komplexnímu chování (WireWord)



1D automaty

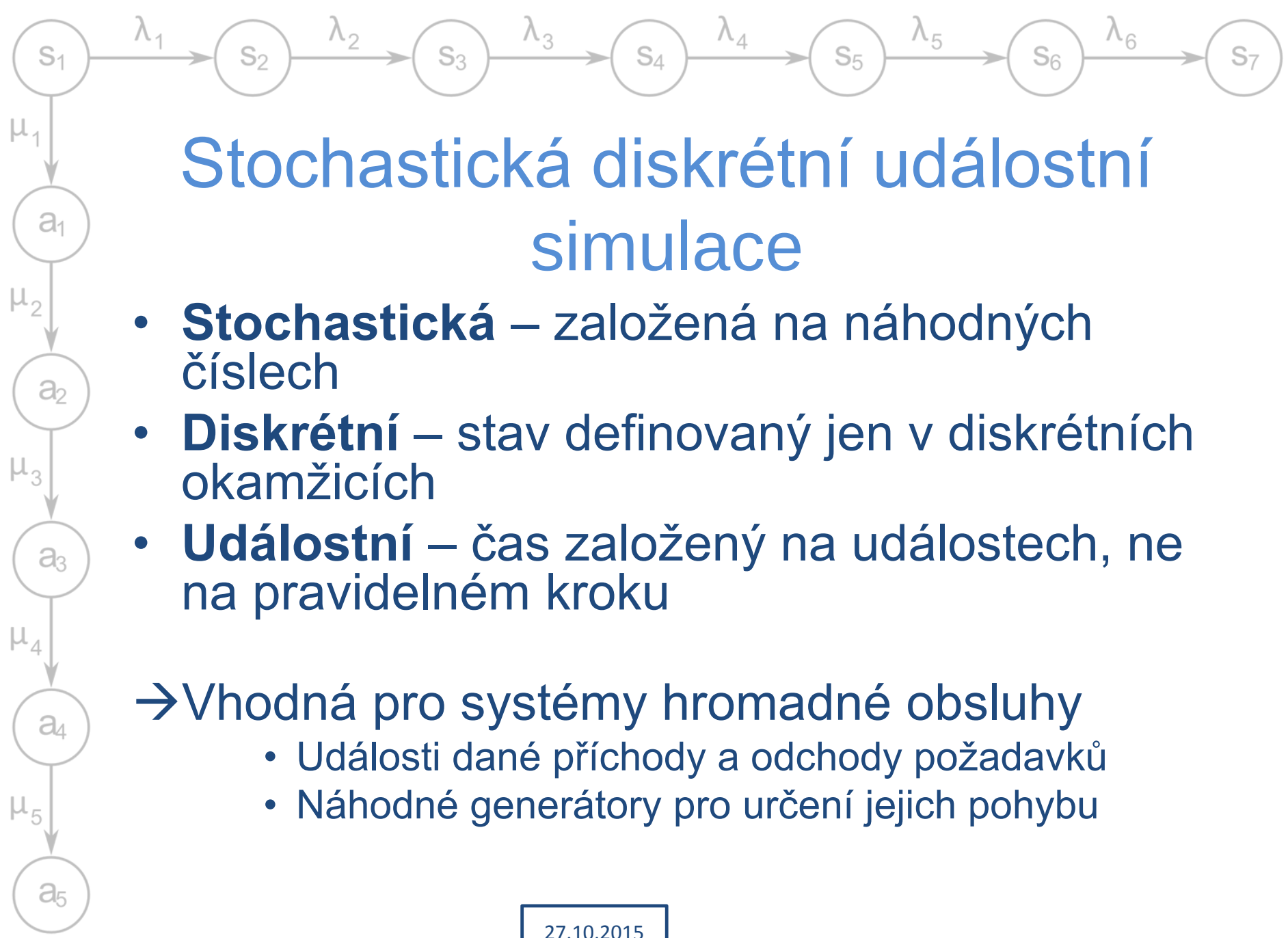
- Systematicky popsány Stephenem Wolframem (1981)
  - Jednoduchý popis ale překvapivě složité chování
- 256 možných pravidel
  - Pravidlo 184 – dopravní simulace

  - Základ pro Nagel-Schreckenbergův model dopravy
- Řada vzorů odpovídacích přírodním nebo matematickým jevům



Netlogo

- Kombinace celulárního automatu (prostředí) a agentů kteří se v něm pohybují
  - Obojí lze volně definovat
- Rychlá tvorba velkého množství simulací
  - Snadné modelování interakcí agentů s prostředím a mezi sebou
- Řada implementací
  - NetLogo (2D), Starlogo (3D)

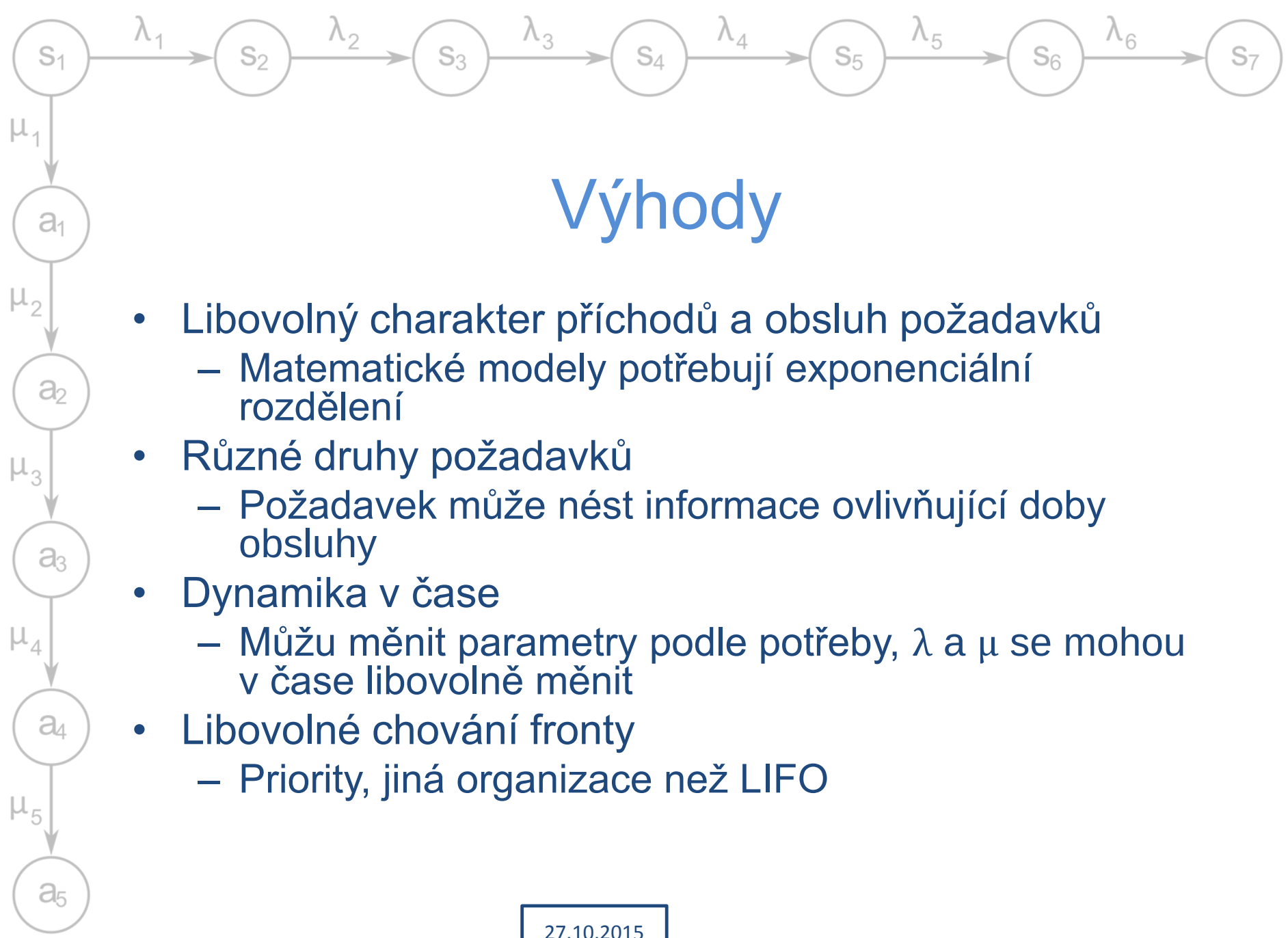


Stochastická diskrétní událostní simulace

- **Stochastická** – založená na náhodných číslech
- **Diskrétní** – stav definovaný jen v diskrétních okamžicích
- **Událostní** – čas založený na událostech, ne na pravidelném kroku

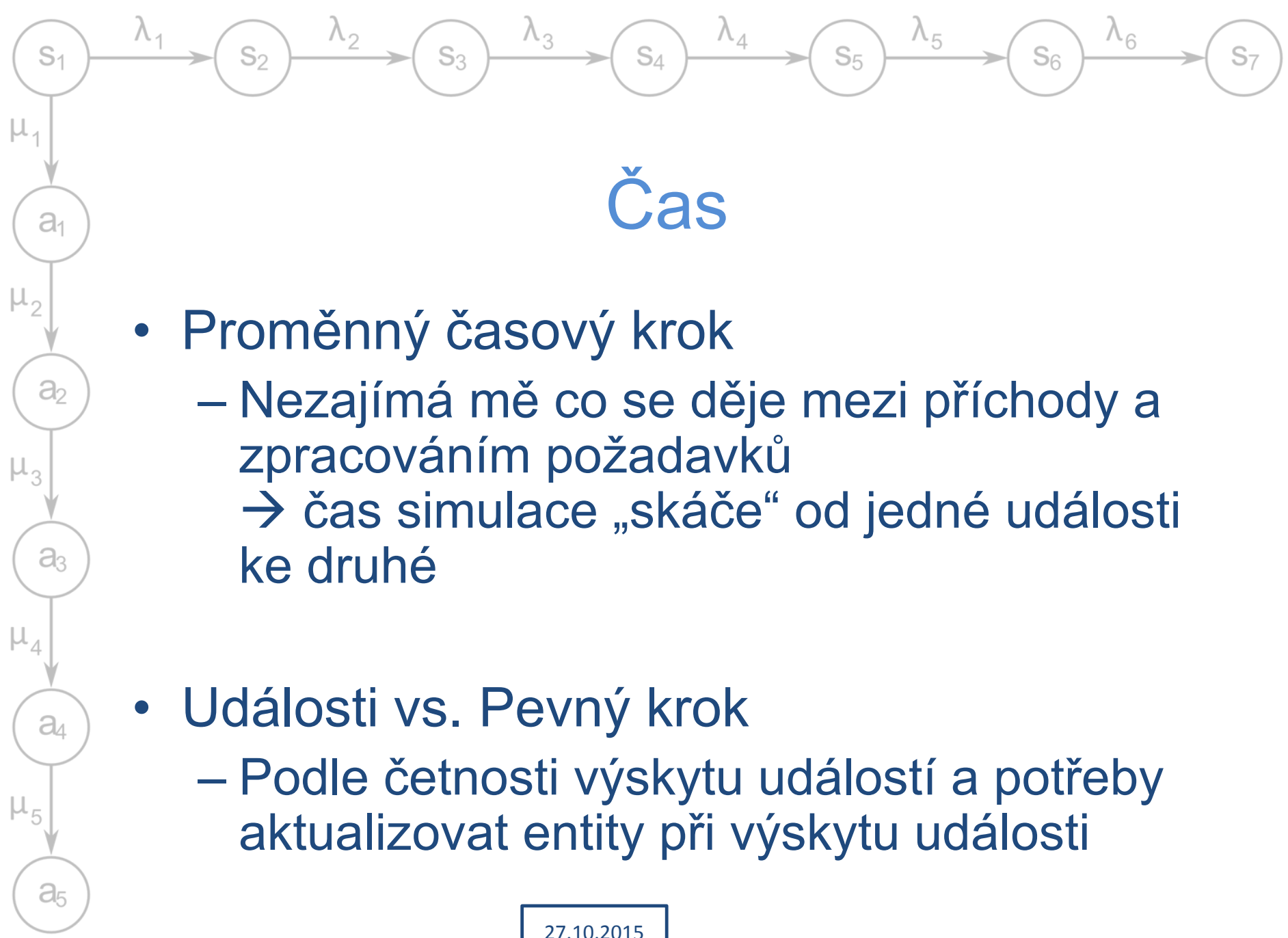
→Vhodná pro systémy hromadné obsluhy

- Události dané příchody a odchody požadavků
- Náhodné generátory pro určení jejich pohybu



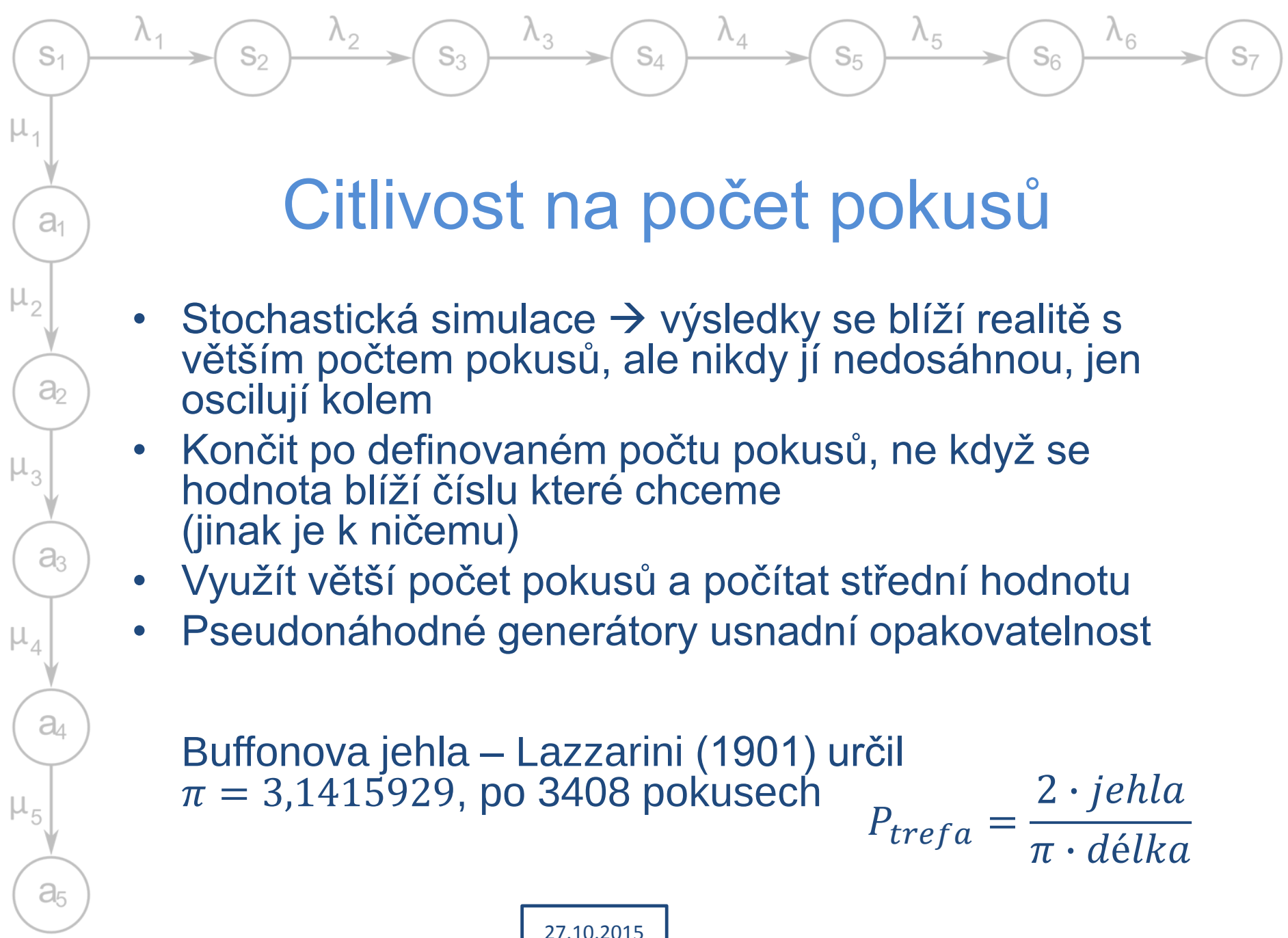
Výhody

- Libovolný charakter příchodů a obsluh požadavků
  - Matematické modely potřebují exponenciální rozdělení
- Různé druhy požadavků
  - Požadavek může nést informace ovlivňující doby obsluhy
- Dynamika v čase
  - Můžu měnit parametry podle potřeby, λ a μ se mohou v čase libovolně měnit
- Libovolné chování fronty
  - Priority, jiná organizace než LIFO



Čas

- Proměnný časový krok
  - Nezajímá mě co se děje mezi příchody a zpracováním požadavků
    → čas simulace „skáče“ od jedné události ke druhé
- Události vs. Pevný krok
  - Podle četnosti výskytu událostí a potřeby aktualizovat entity při výskytu události

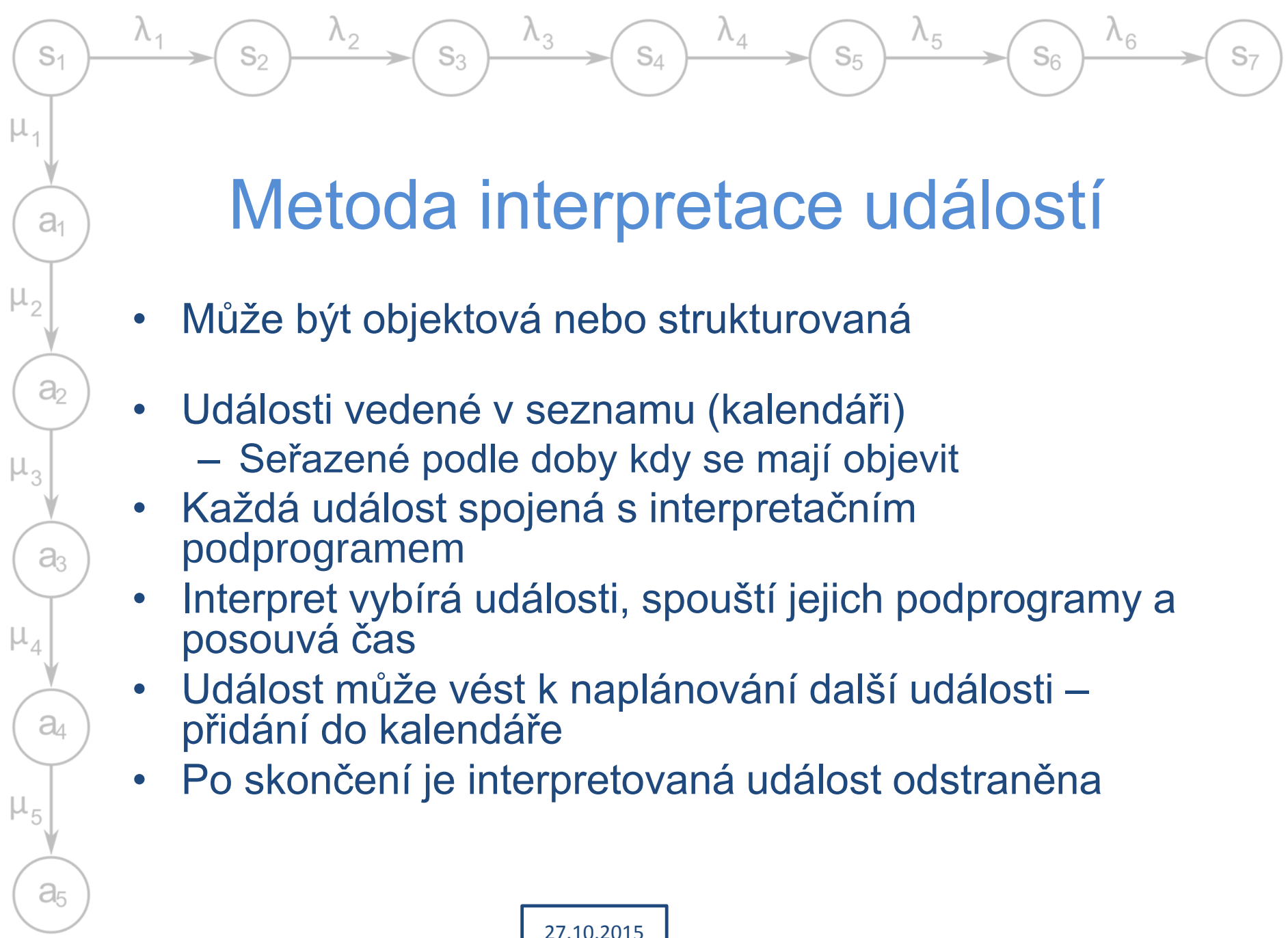


Citlivost na počet pokusů

- Stochastická simulace → výsledky se blíží realitě s větším počtem pokusů, ale nikdy jí nedosáhnou, jen oscilují kolem
- Končit po definovaném počtu pokusů, ne když se hodnota blíží číslu které chceme (jinak je k ničemu)
- Využít větší počet pokusů a počítat střední hodnotu
- Pseudonáhodné generátory usnadní opakovatelnost

Buffonova jehla – Lazzarini (1901) určil $\pi = 3{,}1415929$, po 3408 pokusech

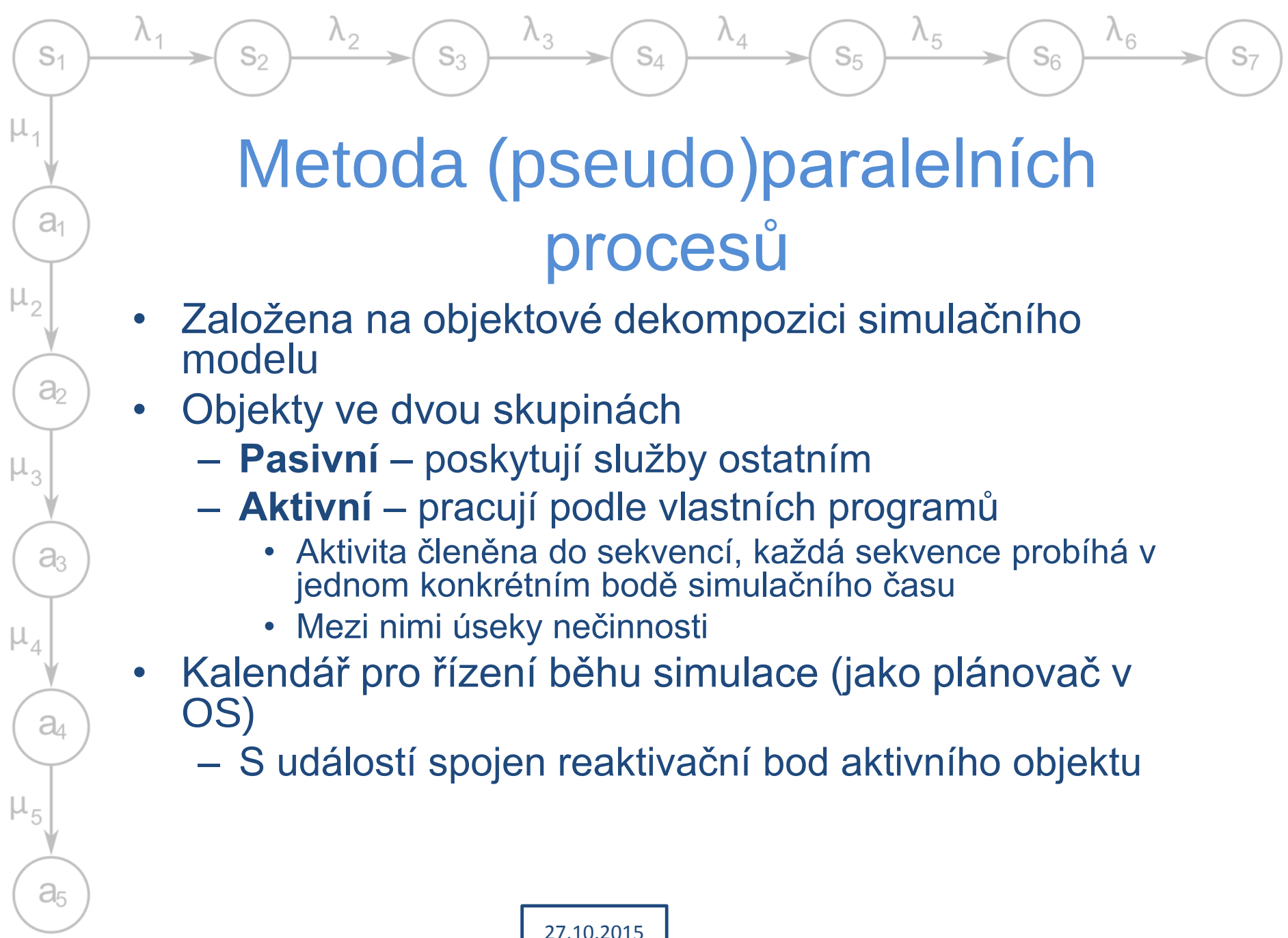
$$P_{trefa} = \frac{2 \cdot jehla}{\pi \cdot délka}$$



Metoda interpretace událostí

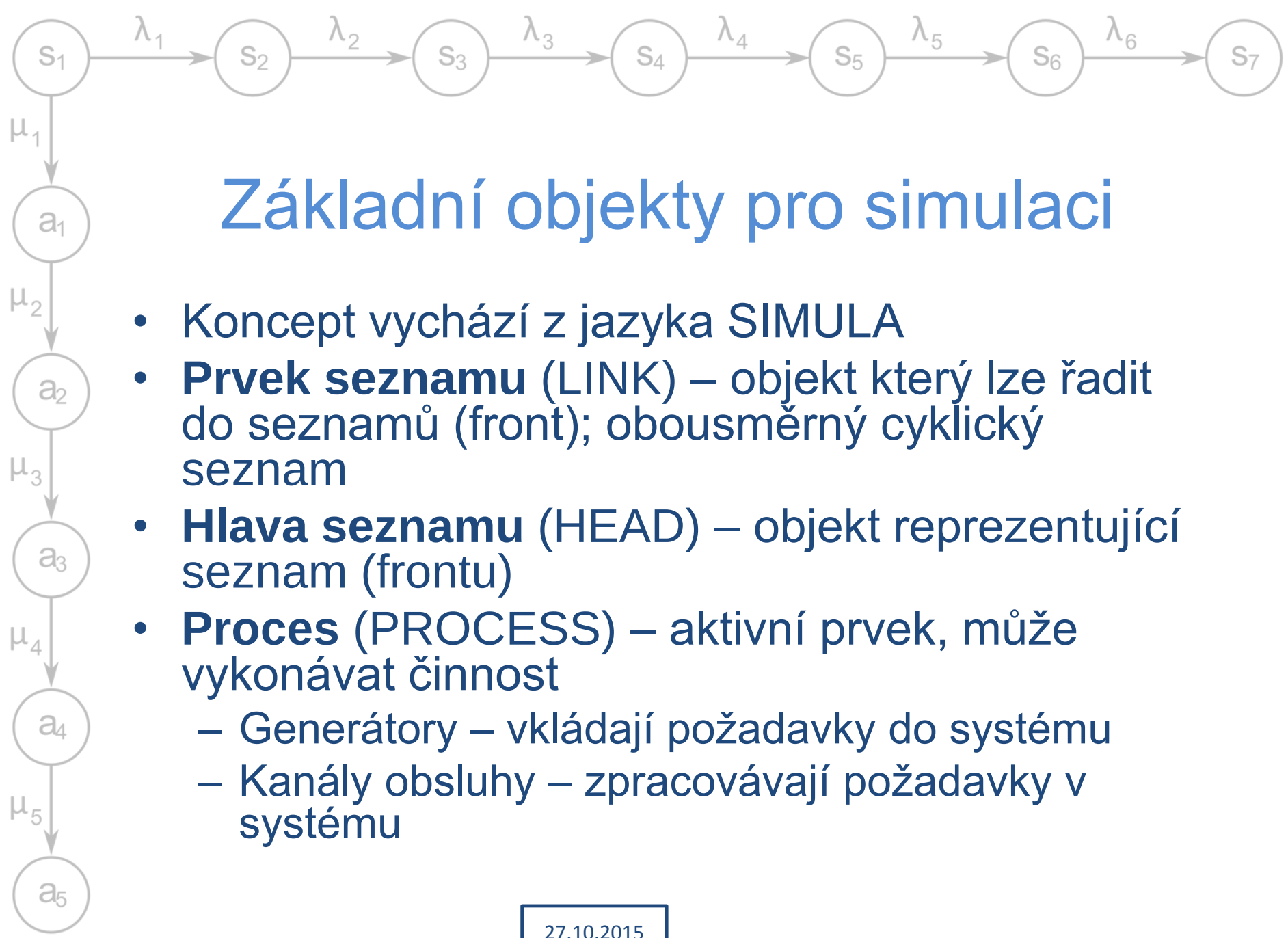
- Může být objektová nebo strukturovaná

- Události vedené v seznamu (kalendáři)
  - Seřazené podle doby kdy se mají objevit
- Každá událost spojená s interpretačním podprogramem
- Interpret vybírá události, spouští jejich podprogramy a posouvá čas
- Událost může vést k naplánování další události – přidání do kalendáře
- Po skončení je interpretovaná událost odstraněna



Metoda (pseudo)paralelních procesů

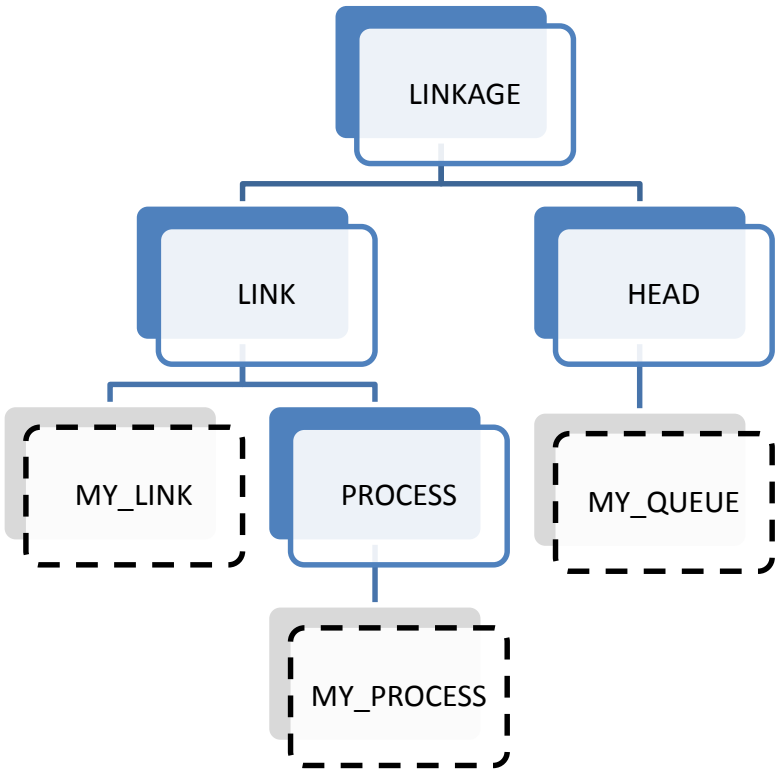
- Založena na objektové dekompozici simulačního modelu
- Objekty ve dvou skupinách
  - **Pasivní** – poskytují služby ostatním
  - **Aktivní** – pracují podle vlastních programů
    - Aktivita členěna do sekvencí, každá sekvence probíhá v jednom konkrétním bodě simulačního času
    - Mezi nimi úseky nečinnosti
- Kalendář pro řízení běhu simulace (jako plánovač v OS)
  - S událostí spojen reaktivační bod aktivního objektu



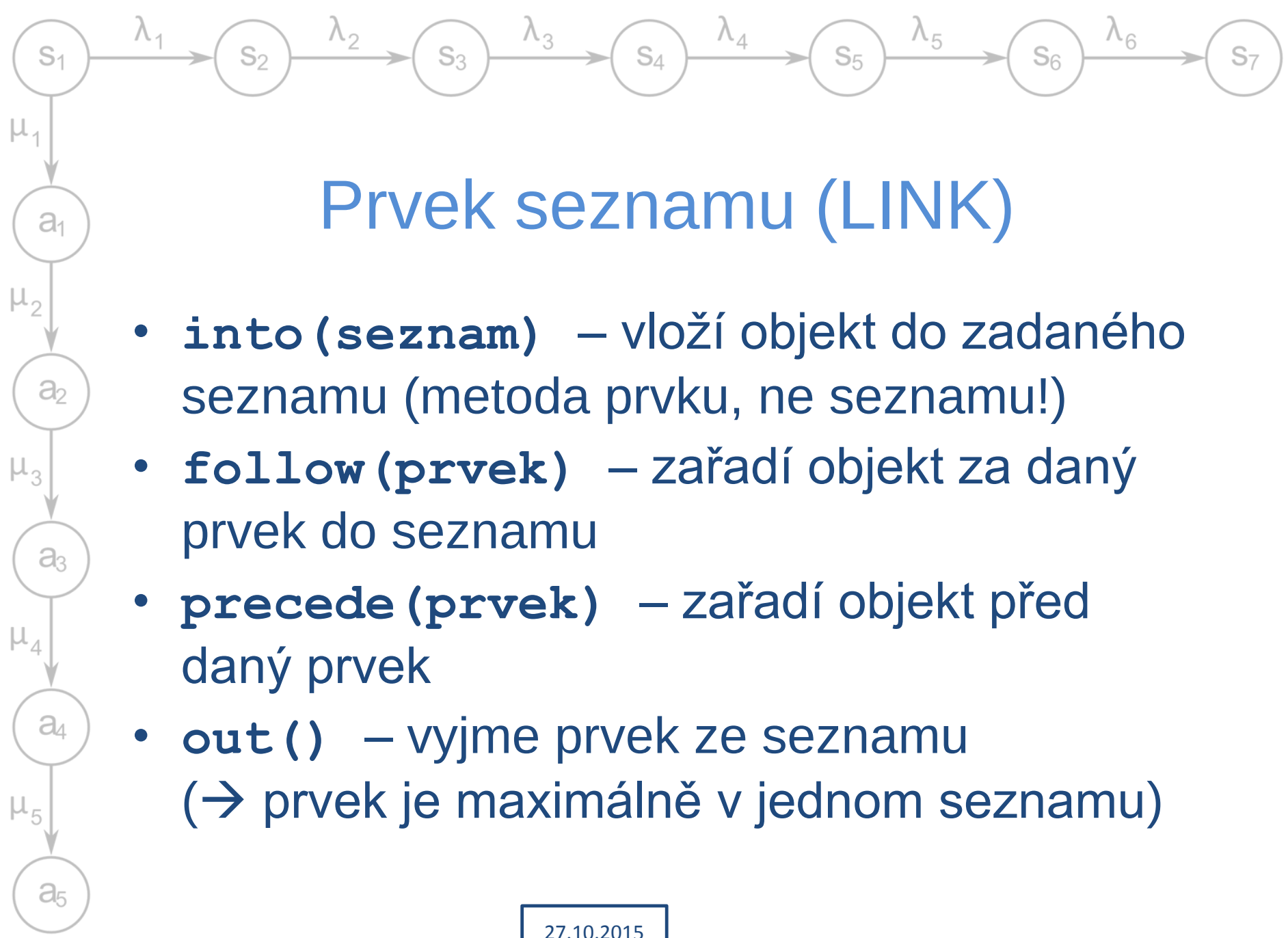
Základní objekty pro simulaci

- Koncept vychází z jazyka SIMULA
- **Prvek seznamu** (LINK) – objekt který lze řadit do seznamů (front); obousměrný cyklický seznam
- **Hlava seznamu** (HEAD) – objekt reprezentující seznam (frontu)
- **Proces** (PROCESS) – aktivní prvek, může vykonávat činnost
  - Generátory – vkládají požadavky do systému
  - Kanály obsluhy – zpracovávají požadavky v systému

Struktura objektů

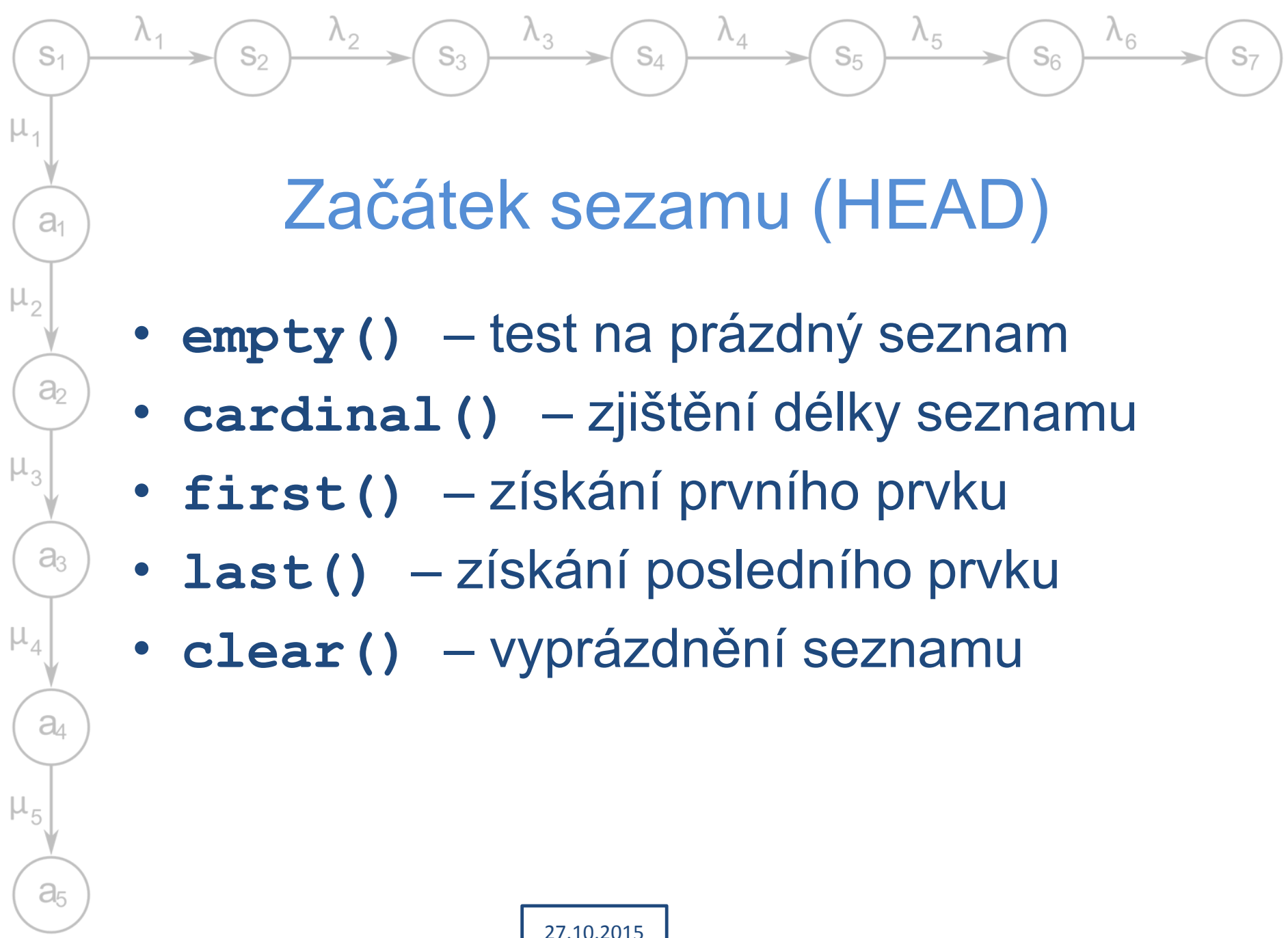


- Procesy je možné řadit do front
  - Kalendář lze chápat jako speciální frontu



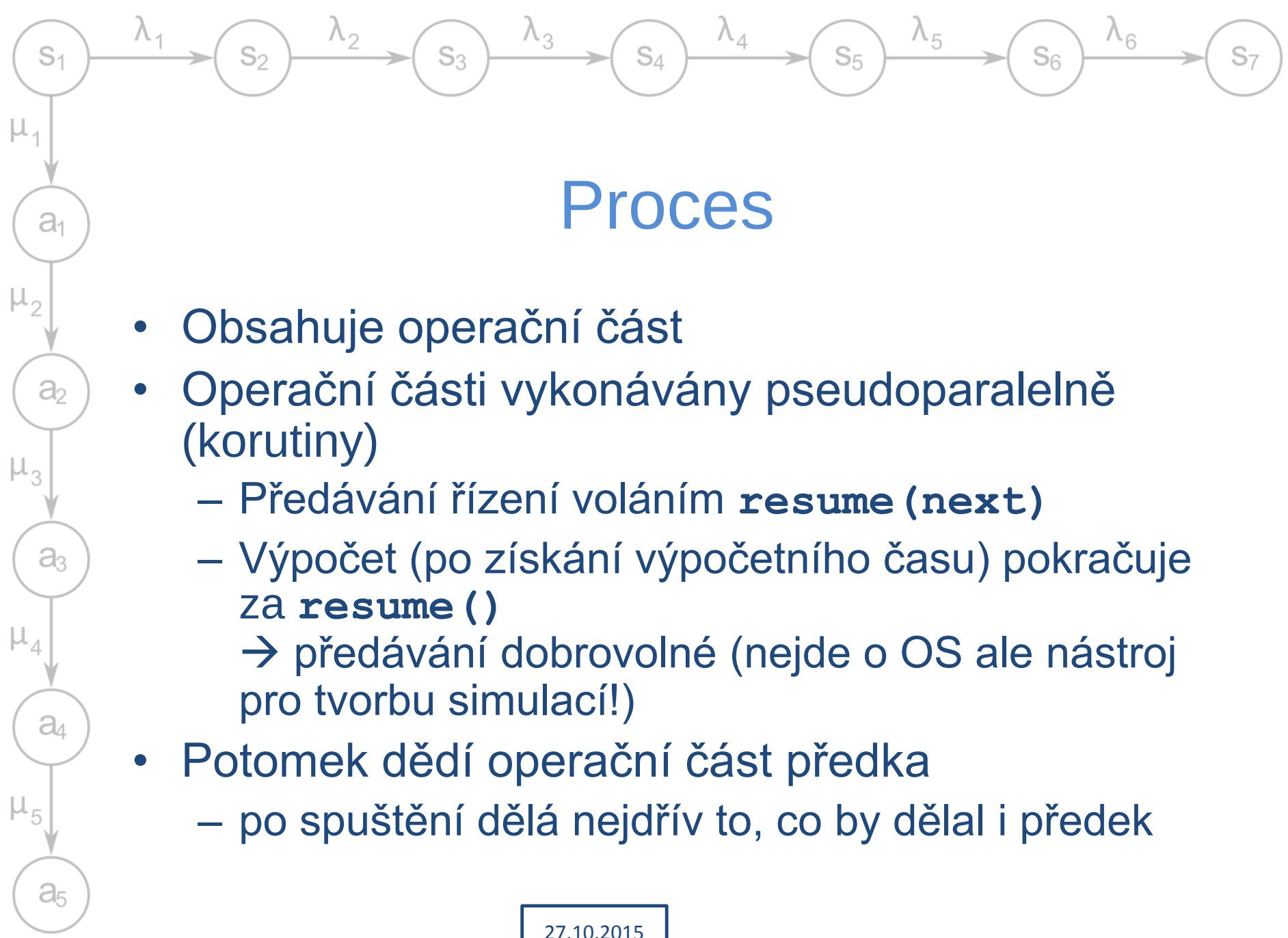
Prvek seznamu (LINK)

- **`into(seznam)`** – vloží objekt do zadaného seznamu (metoda prvku, ne seznamu!)
- **`follow(prvek)`** – zařadí objekt za daný prvek do seznamu
- **`precede(prvek)`** – zařadí objekt před daný prvek
- **`out()`** – vyjme prvek ze seznamu (→ prvek je maximálně v jednom seznamu)



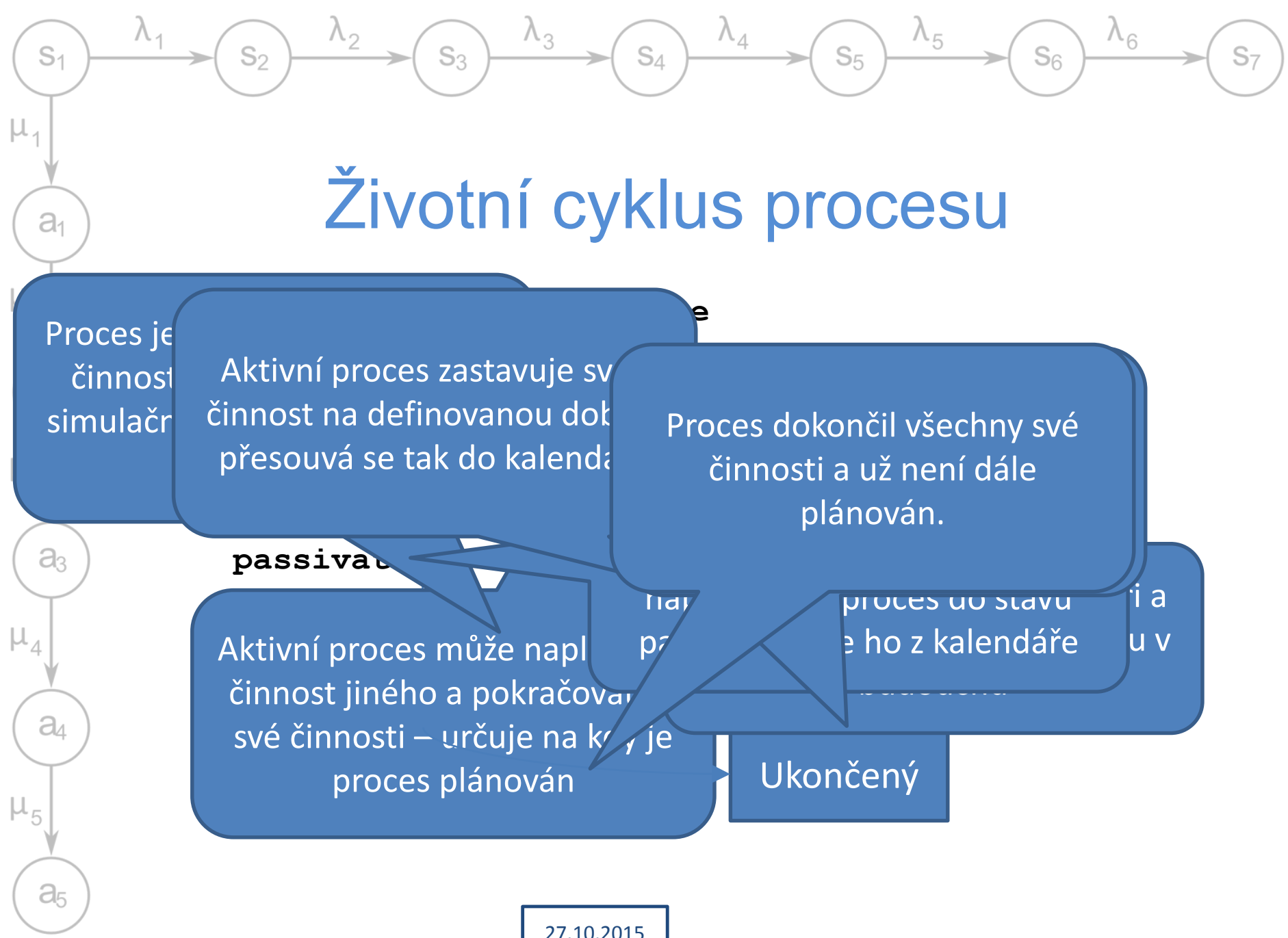
Začátek sezamu (HEAD)

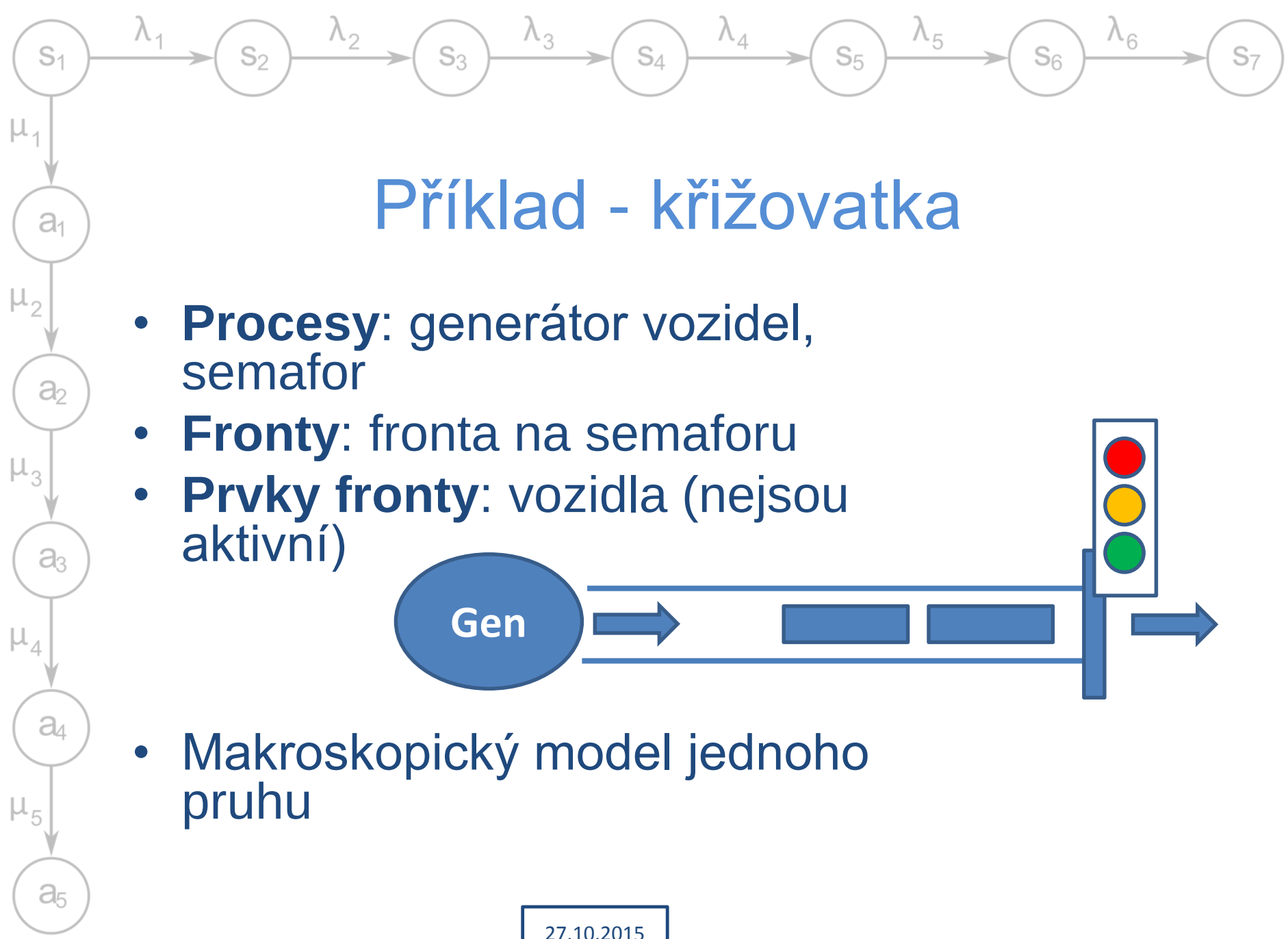
- **`empty()`** – test na prázdný seznam
- **`cardinal()`** – zjištění délky seznamu
- **`first()`** – získání prvního prvku
- **`last()`** – získání posledního prvku
- **`clear()`** – vyprázdnění seznamu



Proces

- Obsahuje operační část
- Operační části vykonávány pseudoparalelně (korutiny)
  - Předávání řízení voláním **`resume(next)`**
  - Výpočet (po získání výpočetního času) pokračuje za **`resume()`**
    → předávání dobrovolné (nejde o OS ale nástroj pro tvorbu simulací!)
- Potomek dědí operační část předka
  - po spuštění dělá nejdřív to, co by dělal i předek

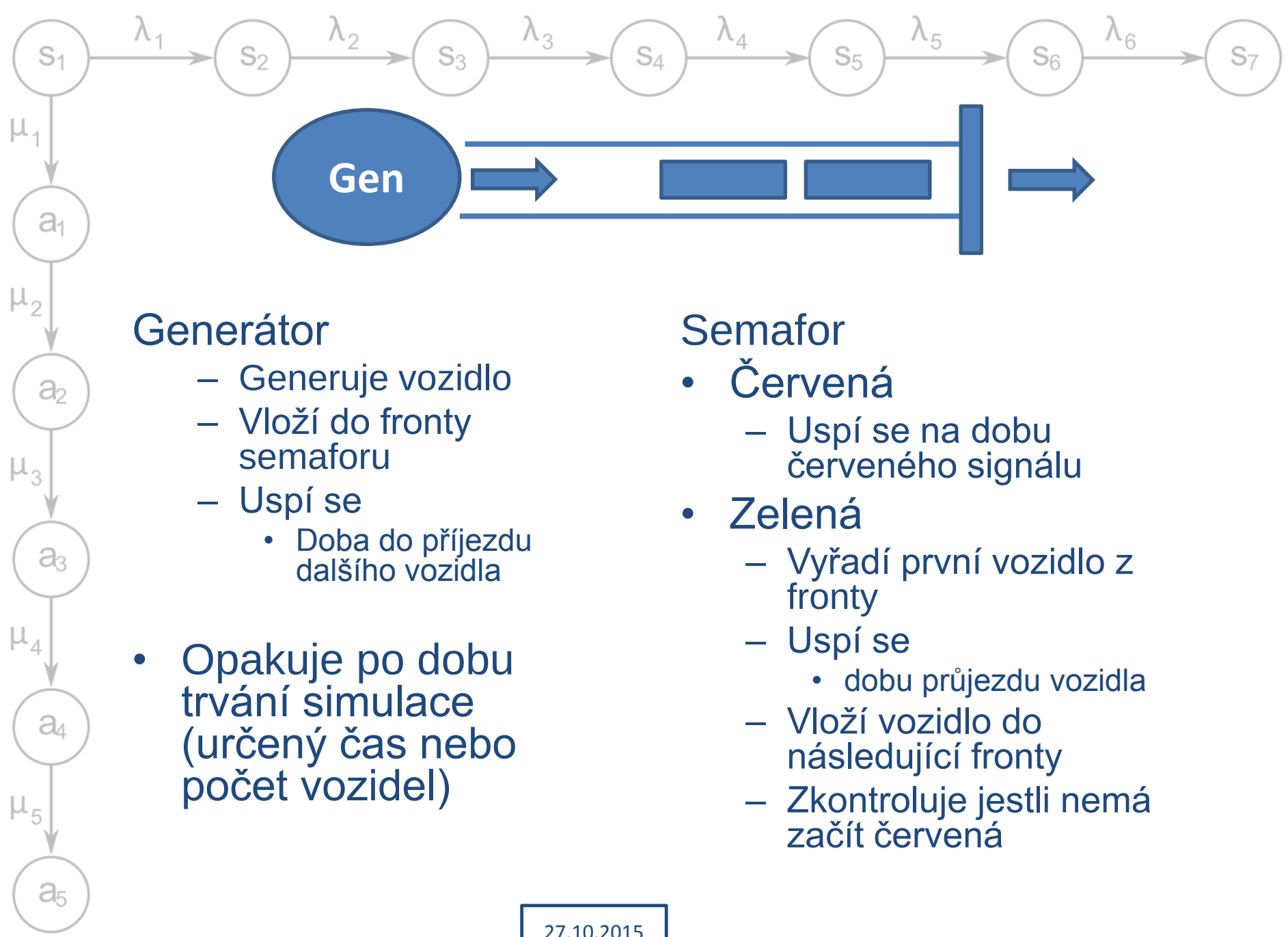

Životní cyklus procesu
Proces je
činnost
simulačr
Aktivní proces zastavuje sv
činnost na definovanou dob
přesouvá se tak do kalenda
Proces dokončil všechny své činnosti a už není dále plánován.
passivat
Aktivní proces může napl
činnost jiného a pokračova
své činnosti – určuje na ko
proces plánován
Ukončený
27.10.2015



Příklad - křižovatka

- **Procesy**: generátor vozidel, semafor
- **Fronty**: fronta na semaforu
- **Prvky fronty**: vozidla (nejsou aktivní)

- Makroskopický model jednoho pruhu



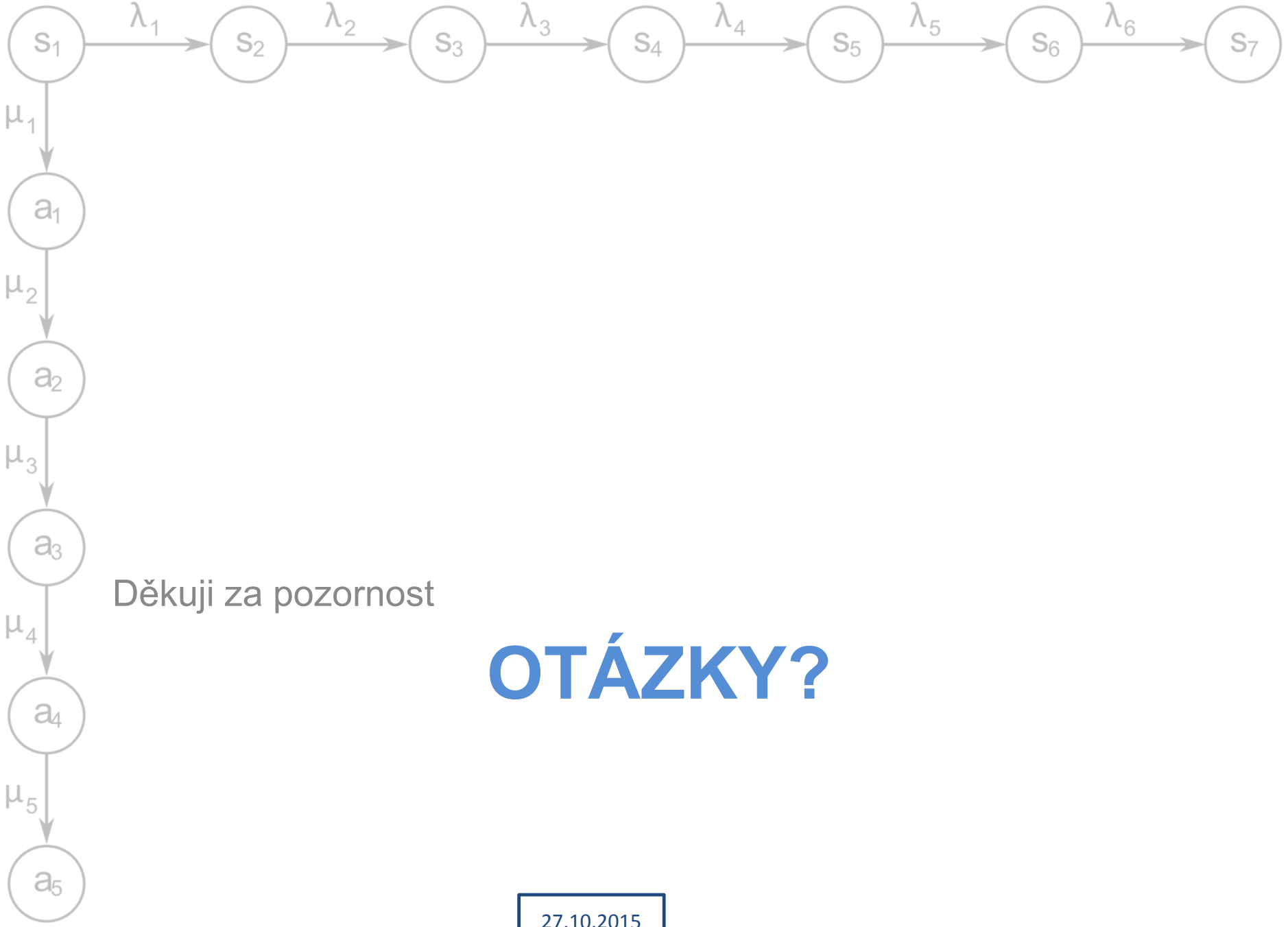
Generátor

- Generuje vozidlo
- Vloží do fronty semaforu
- Uspí se
  - Doba do příjezdu dalšího vozidla

- Opakuje po dobu trvání simulace (určený čas nebo počet vozidel)

Semafor

- Červená
  - Uspí se na dobu červeného signálu
- Zelená
  - Vyřadí první vozidlo z fronty
  - Uspí se
    - dobu průjezdu vozidla
  - Vloží vozidlo do následující fronty
  - Zkontroluje jestli nemá začít červená



Děkuji za pozornost

OTÁZKY?