JANOG45


# Declarative Networking

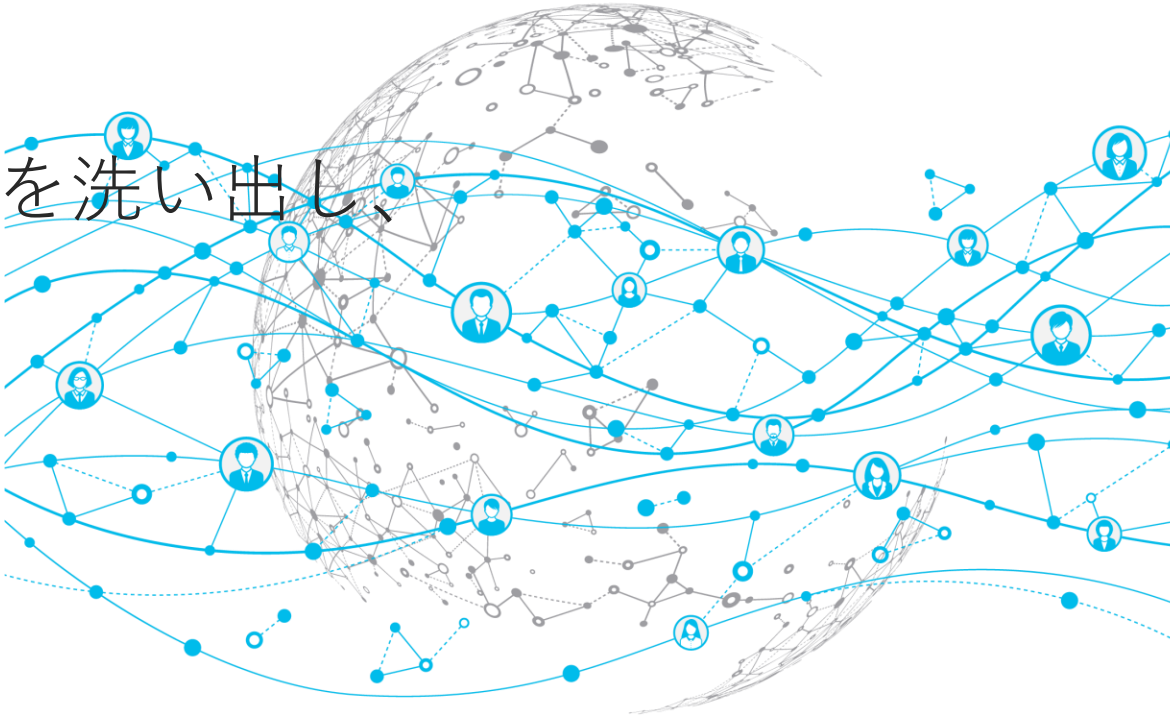
## 基礎理論、実例、今後の展望

Miya Kohno (mkohno@cisco.com)

22 January 2020

# はじめに

- ネットワークは複雑で不確定性の高い分散システム
- 本セッションで達成したいこと
  - 「ネットワークシステムを制御するためには、Declarativeな手法が適している」という仮説導出とその例証
  - 「そうは言っても..」という課題を洗い出し、議論と今後の展望



# Declarative(宣言的)とは

- Imperative(命令的)・Procedural(手続き的) との 対義的概念
- 命令的に やり方(how) を示すのでなく、目的や結果(what, desired state) を合意する
- 数理論理学 に基づき、モデル駆動形 とも親和性が高い [*1]
- desired stateを合意するため、冪等性 [*2]にもつながる

[*1] “A language with a clear correspondence to mathematical logic”
(http://en.wikipedia.org/wiki/Declarative_programming)

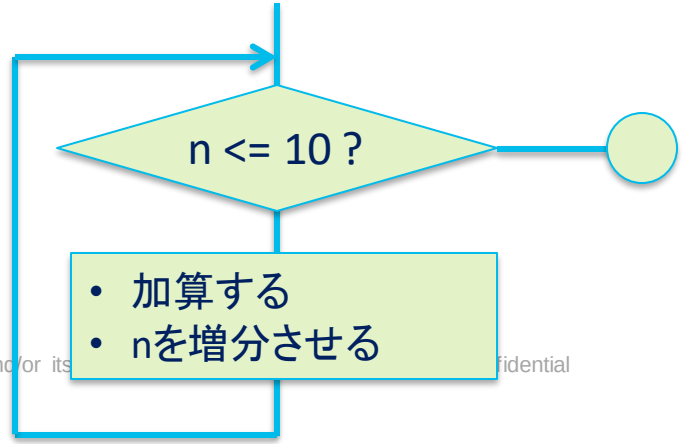
[*2] ある操作を1回行っても数回行っても結果は等しいこと, Idempotence

# Declarative Programming（宣言的プログラミング）

「1から10までの全ての整数を足す」

## Procedure/Imperative

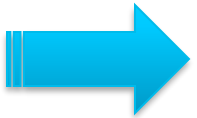
```
var s = 0;
for(var n=1; n<=10; n++)
 {
   s = s + n;
 }
console.log(s);
//55
```




## Declarative

```
(->> (range 1 11)
     (reduce +)
     (println)
)
//55
```

1から10という範囲の整数の集合

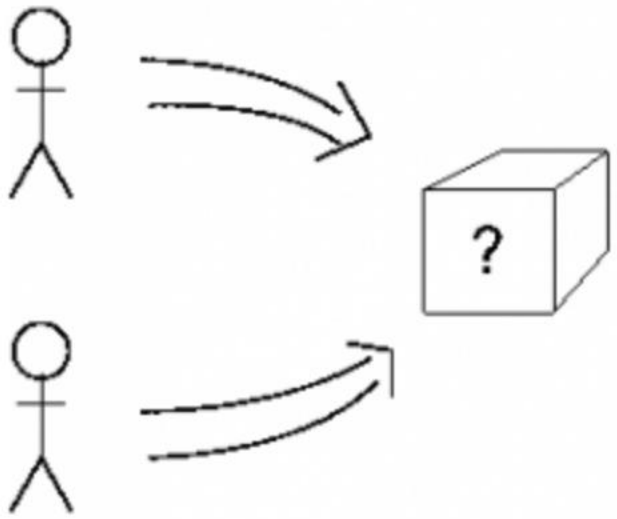
Desired State
“総計”




http://karari.tumblr.com/post/61067682037/clojure

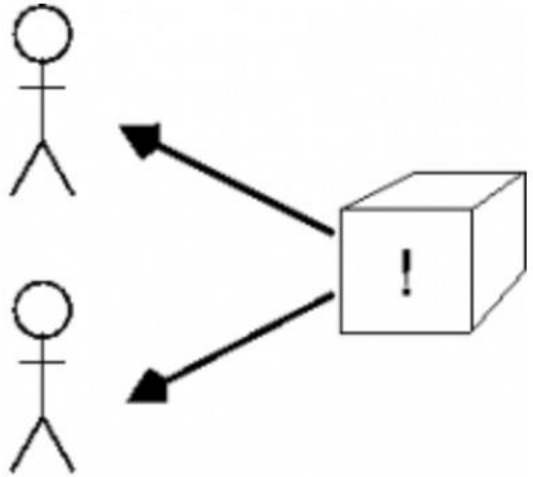
# Declarative の強み

## Procedure/Imperative




Obligation view

## Declarative




Promise view

<u>逐一指示するモデル:</u>

指示の一部が失われたり、Ackが届かなかったり、処理に誤りがあったりすると、指示の意図と結果が異なる可能性がある。

<u>意図する結果を予め合意するモデル:</u>

方法の指示でなく、結果を合意しているため、不確定性の高い環境においても頑健である。

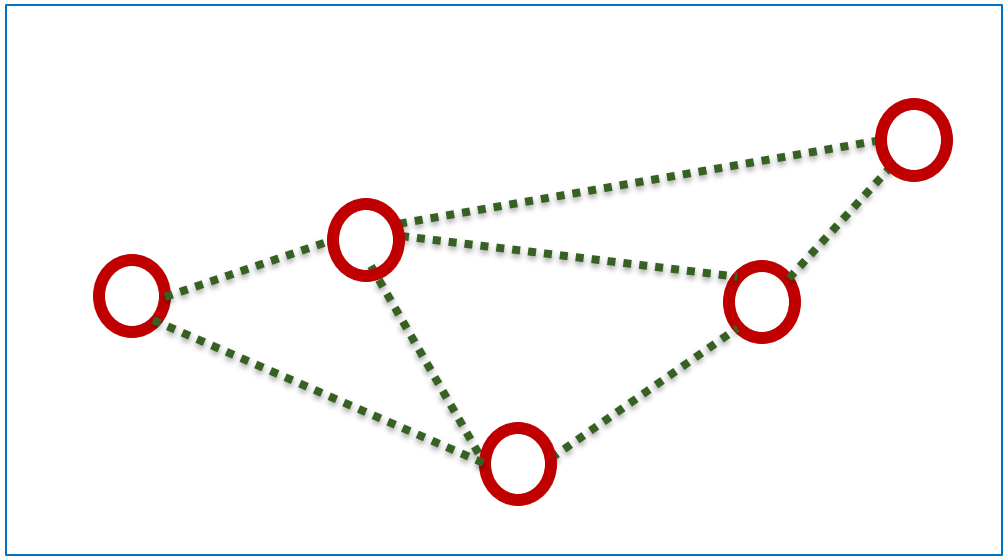


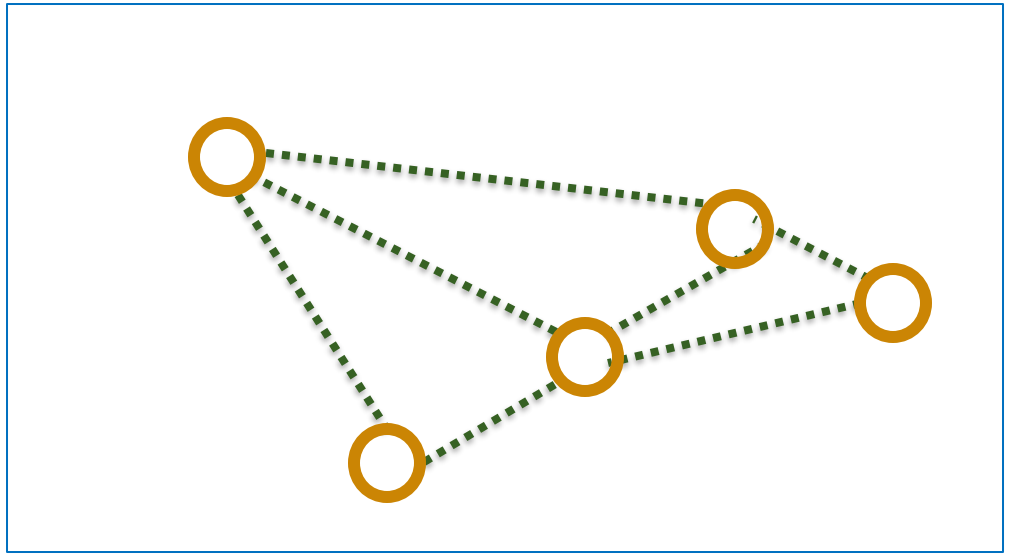
https://www.linuxjournal.com/content/promise-theory%E2%80%94what-it

# Declarative Networking の例

- Routing Protocols
- Group Based Policy
- Cloud Native API
- Network Data Log (NDlog)
- SRv6 Network Programming

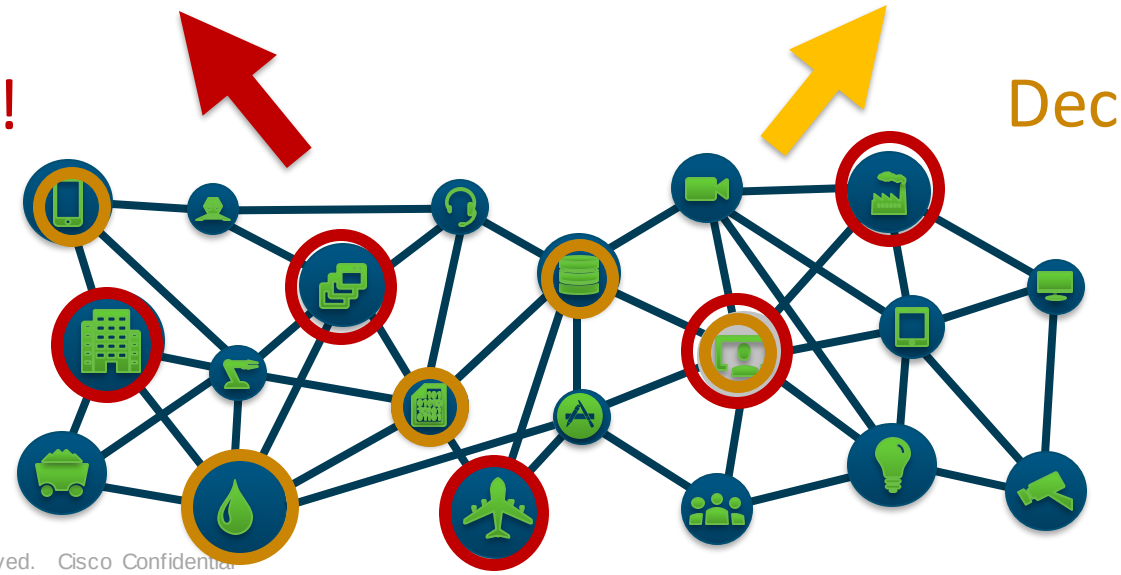
# Routing Protocol は Declarative !





Declare **Red** !

Declare **Yellow** !!



- Opaque LSA (OSPF)
- Optional Transitive (BGP)

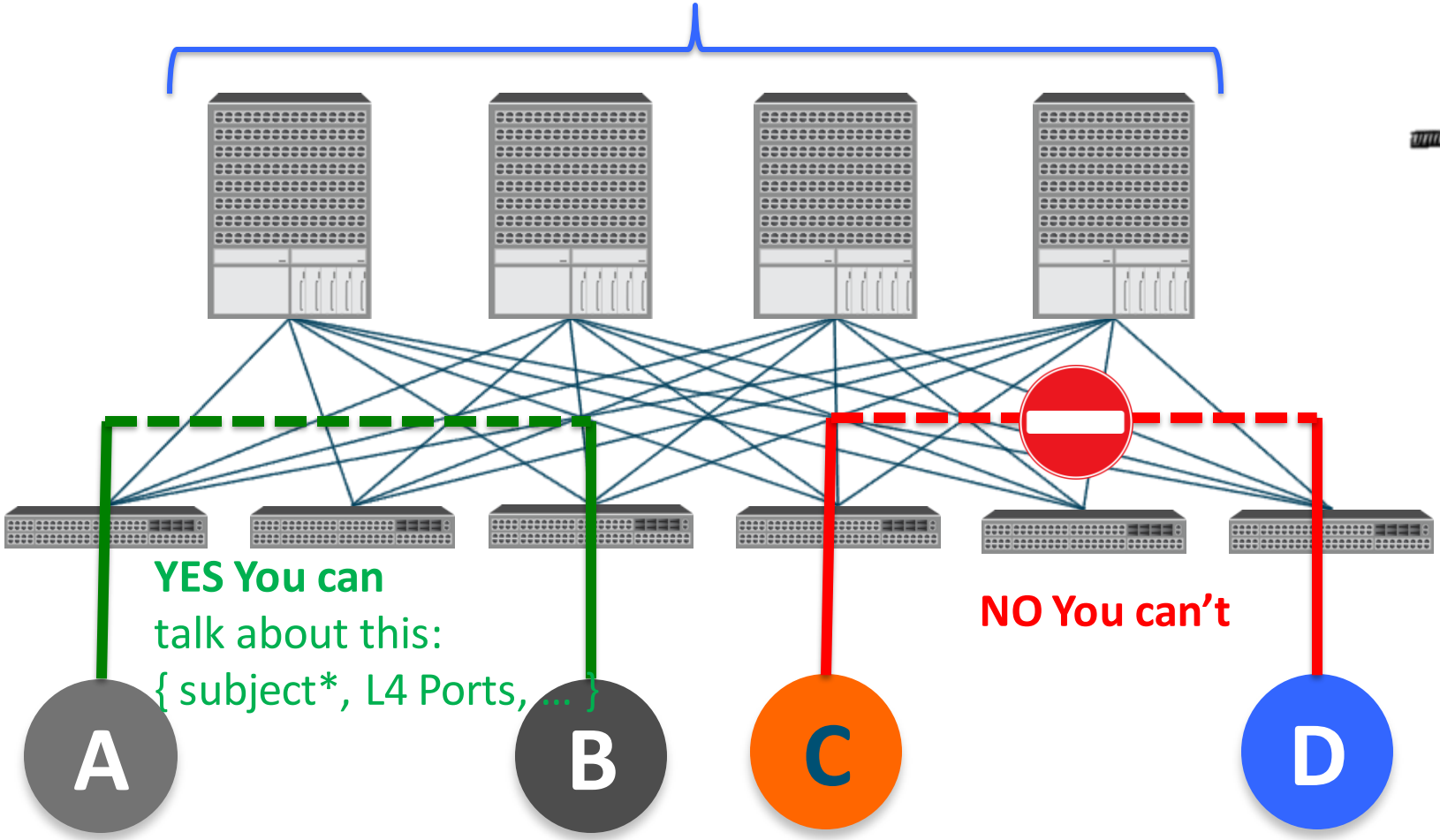


このおかげで、複雑大規模なインターネットは進化しながら今日も動いている

# Group Based Policy

## Network Controlって何？

→ the rest is path optimization

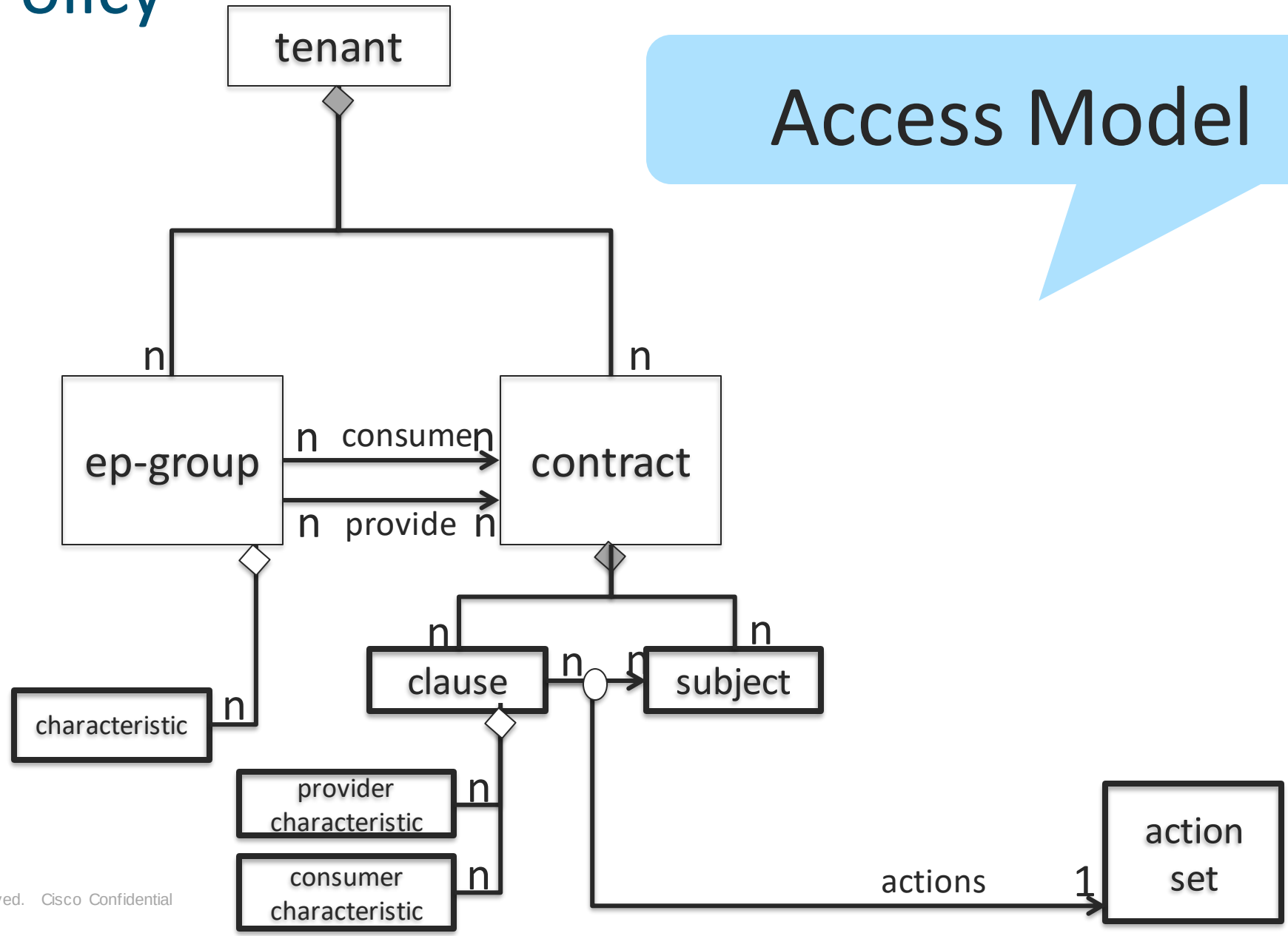



→ End point A can talk to end point B

→ End point C can’t talk to end point D

Source : Mike Dvorkin

# Group Based Policy




Source : Mike Dvorkin

# Cloud Native API

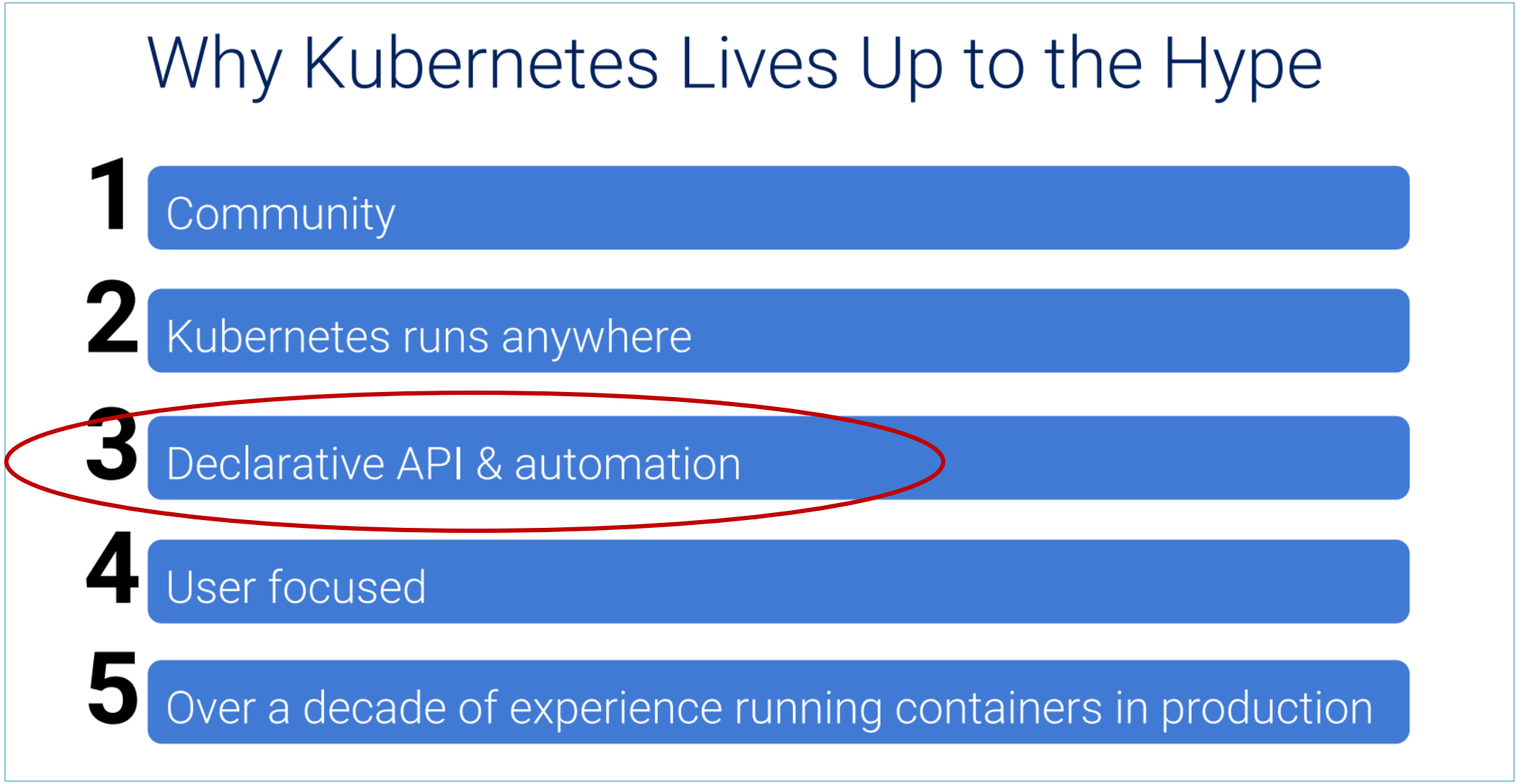

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and **declarative APIs** exemplify this approach. …

クラウドネイティブ技術により、組織はパブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの最新の動的環境で、スケーラブルなアプリケーションを構築および実行可能になる。コンテナ、サービスメッシュ、マイクロサービス、不変のインフラストラクチャ、および**宣言型API**は、このアプローチの例である。（後略）



https://github.com/cncf/toc/blob/master/DEFINITION.md

# Cloud Native API






https://kccna18.sched.com/event/Gsxq/keynote-kubernetes-living-up-to-the-hype-janet-kuo-software-engineer-google

# Network Data Log (Ndlog)

## Declarative Networking

Boon Thau Loo  Tyson Condie  Minos Garofalakis  David E. Gay
Joseph M. Hellerstein  Petros Maniatis  Raghu Ramakrishnan
Timothy Roscoe  Ion Stoica

*University of California-Berkeley*  *University of Pennsylvania*  *Intel Research Berkeley*
*ETH Zurich*  *Yahoo! Research*  *Technical University of Crete*

boonloo@cis.upenn.edu  {tcondie,hellerstein,istoica}@cs.berkeley.edu  minos@softnet.tuc.gr
{david.e.gay,petros.maniatis}@intel.com  ramakris@yahoo-inc.com  troscoe@inf.ethz.ch

**ABSTRACT**

Declarative Networking is a programming methodology that enables developers to concisely specify network protocols and services, which are directly compiled to a dataflow framework that executes the specifications. This paper provides an introduction to basic issues in declarative networking, including language design, optimization and dataflow execution. We present the intuition behind declarative programming of networks, including roots in Datalog, extensions for networked environments, and the semantics of long-running queries over network state. We focus on a sublanguage we call Network Datalog (*NDlog*), including execution strategies that provide crisp eventual consistency semantics with significant flexibility in execution. We also describe a more general language called *Overlog*, which makes some compromises between expressive richness and semantic guarantees.

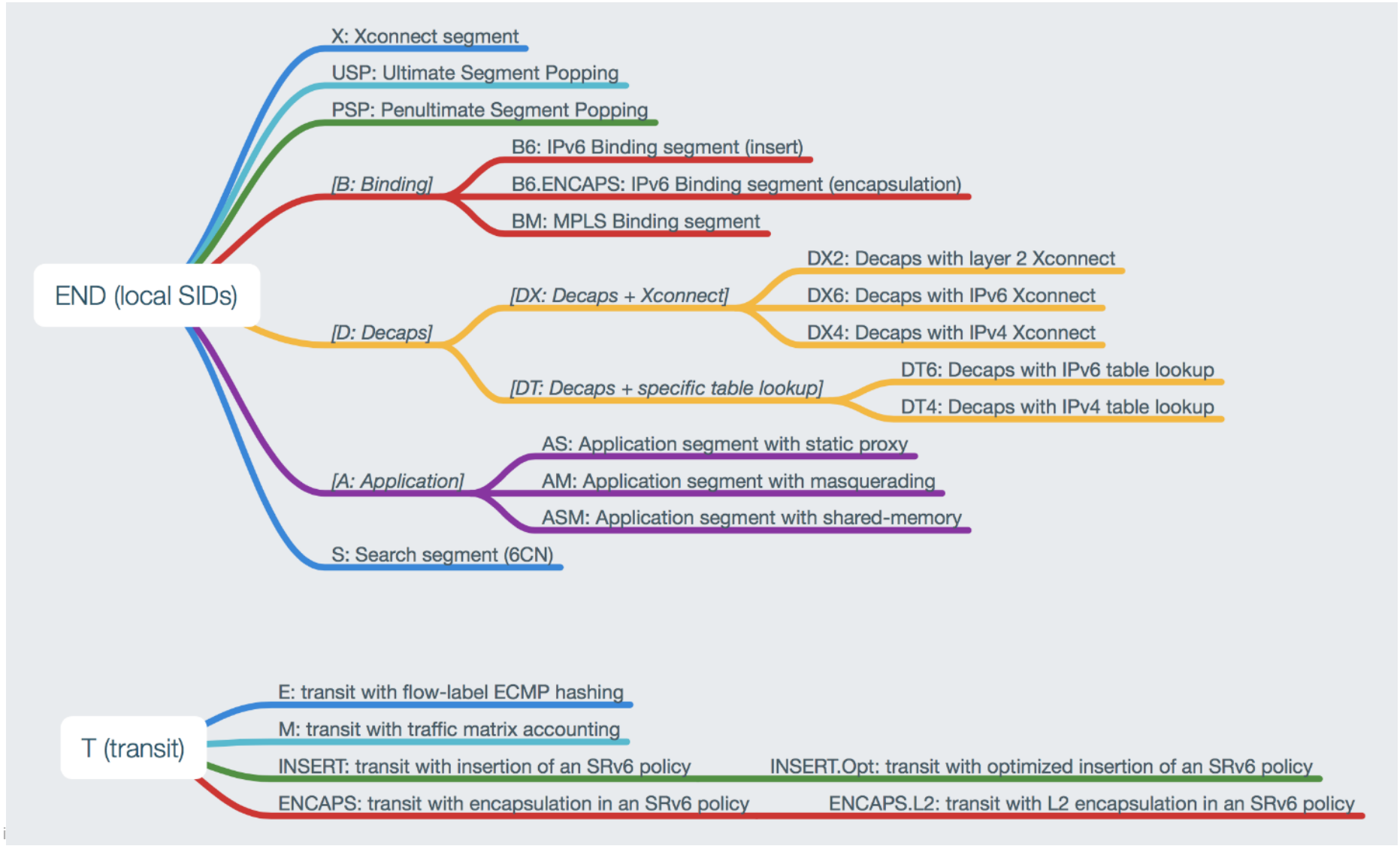
protocol implemented correctly is even harder. And, in order to change or upgrade a deployed routing protocol today, one must get access to *each* router to modify its software. This process is made even more tedious and error-prone by the use of conventional programming languages.

In this paper, we introduce *declarative networking*, an application of database query-language and processing techniques to the domain of networking. Declarative networking is based on the observation that network protocols deal at their core with computing and maintaining distributed state (e.g., routes, sessions, performance statistics) according to basic information locally available at each node (e.g., neighbor tables, link measurements, local clocks) while enforcing constraints such as local routing policies. Recursive query languages studied in the deductive database literature [27] are a natural fit for expressing the relationship between base data, derived data, and the associated constraints. As we



https://dl.acm.org/doi/10.1145/1592761.1592785

# SRv6 Network Programming - Functions






https://p4.org/assets/p4-ws-2017-the-extensible-network.pd

# そうは言っても… 論理の限界

論理は、現実的には結構脆弱

- 「ピザ一枚2,400円だそうです。15,000円あれば、ピザ5枚買えますね！」

  - 論理的には全く”真” ( 2,400 * 5 <= 15,000 ) → T

  -しかし実際は、ここは突っ込まないと…

  「いや、6枚買えるでしょう」

# そうは言っても… 数理の限界

| 方程式 | 線形方程式 | | | 非線形方程式 | | |
|---|---|---|---|---|---|---|
| | 1個の方程式 | 数個の方程式 | 多数の方程式 | 1個の方程式 | 数個の方程式 | 多数の方程式 |
| 代数 | ごく容易 | 容易 | 事実上不可能 | 非常に困難 | 非常に困難 | 不可能 |
| 常微分 | 容易 | 困難 | 事実上不可能 | 非常に困難 | 不可能 | 不可能 |
| 偏微分 | 困難 | 事実上不可能 | 不可能 | 不可能 | 不可能 | 不可能 |

Source : Mathematical Modeling in Chemical Engineering,
Cambridge University Press (2014/3/20)

# そうは言っても…

- ドメイン固有[*]ならよいが、汎用的なきめ細かいことはやりにくい。。。
- モデルとトランザクションだけでは、自動化できない。やっぱりScriptやWorkflowは必要。。。

[*] Domain Specific Language (DSL) ：
宣言的プログラミング言語の一種とされる。
領域を限定することにより”What/Desired State”を合意しやすい。
SQLが代表例。Routing Protocol, Network ProgrammingもDomain Specificと考えられる。

# Declarative Networking - 今後の展望

- 「しかしながら、可能な限りこの方法に従うことによって、プログラムの信頼性は大きく向上する」

  Richard Bird著、山下伸夫訳、関数プログラミング入門 Haskellで学ぶ原理と技法

- 機械学習や確率的捉え方によって、ある程度数理を補強することはできる

→ 特にネットワークシステムの制御に関しては、
可能な限り、Imperative(命令的)・Procedural(手続き的)
手法を排除しよう！

# 補足資料

- 宣言的ネットワーキングとシステム理論について

  「システム理論の続き – 宣言的ネットワーキング」
  https://qiita.com/mkohno/items/ac83cf461632fbf4d862
  Dec. 2019, Cisco Advent Calendar

- 分散システムと整合性の問題

  「Network Programmabilityとステート性・トランザクション性」
  https://www.mpls.jp/2015/presentations/MK_state-in-programming_01.pdf
  Nov. 2015, MPLS Japan https://www.mpls.jp/2015/index.html


CISCO