ソフトウェア工房
アート，理学，工学の接点

上田 和紀
早稲田大学基幹理工学部情報理工学科
2026年3月14日

最終講義メニュー

- 生い立ち～大学時代 (1956～1983)
- 早稲田大学着任まで (1983～1993)
- 早稲田大学着任後 (1993～2026)
  - 研究
  - 教育
  - 研究室は工房？
  - 国際協力
- 学会活動

全部は話せません．言及もれはどうかご許しください

記念品について

- 業務用硬券（昔のきっぷ）印刷機をもつ業者にオーダー
  - 板紙も業務用
  - 凸版印刷
  - 日付印字機は北陸線加賀温泉駅で使われていたもの
  - 入鋏は懇親会で

係員に渡さずにお持ち帰りください

【備考】名札は山本直輝講師の発案・デザインです

生い立ち

- 1956年東京生まれ
  - 母方の祖父は当時岩見沢駅長
  - 父方の祖父は戦中まで新宿保線区長（鐵道省）
  - 社宅４階の出窓から中央線の電車・列車を見て育つ
- 1963年夏
  - 東海道新幹線試乗（鴨宮実験線）
    - 1000形1004
  - なぜこの話をするのかは後ほど

https://noble-joker.com/nj_home/@1okuno/kamonomiya/

大学時代・コンピュータとの出会い

- 1975年秋学期
  - 東大駒場「算法通論」(FORTRAN)
  - 初回授業担当が筧捷彦先生（当時立教大学）
  - 本郷の教育用計算機センターに通い詰める
    - パンチカード，ラインプリンタ
    - 国鉄スト権ストの休講期間中は毎日
  - 当時のNHK教育テレビ「コンピューター講座」
    島内剛一(立教大)，石田晴久(東大)，
    筧捷彦(立教大)，木村泉(東工大)，
    廣瀬健(早大)，米田信夫(学習院大→東大)

学部後期課程

- 1976年 工学部計数工学科（名称に惹かれる）
  - 卒研配属は学生主導，くじで優先順位決定
    - 教授陣は待っていればよい
  - 暗黙ルール：進学者は別研究室で卒業研究
    - 多重分光画像解析 ― 永田町電総研でのアルバイト（「大津の二値化法」プログラマ）に役立つ
- 1978年

大学院

- 1978年 工学系研究科情報工学専門課程
  和田英一研究室

https://www.s.u-tokyo.ac.jp/ja/info/11015/
東大小柴ホール，2025年11月

Happy Hacking Keyboard
（初代，1999年製，
キートップ換装，現役）

おっと …

- 1978年 工学系研究科情報工学専門課程
  和田英一研究室

Happy Hacking Keyboard
（初代，1999年製，
キートップ換装，現役）

IPSJ Magazine

［巻頭コラム］

計算機科学の終焉

■和田 英一

「情報処理」2025年11月号

さて，どっちか …

IPSJ Magazine

［巻頭コラム］

計算機科学は永遠

■萩谷 昌己

「情報処理」2026年1月号

IPSJ Magazine

［巻頭コラム］

計算機科学の終焉

■和田 英一

「情報処理」2025年11月号

和田研での生活が始まる

- 「今年も面白いことやりましょう」（1978年年賀状）
- モットーは
  「梁山泊」（侃々諤々）＋「門前の小僧」

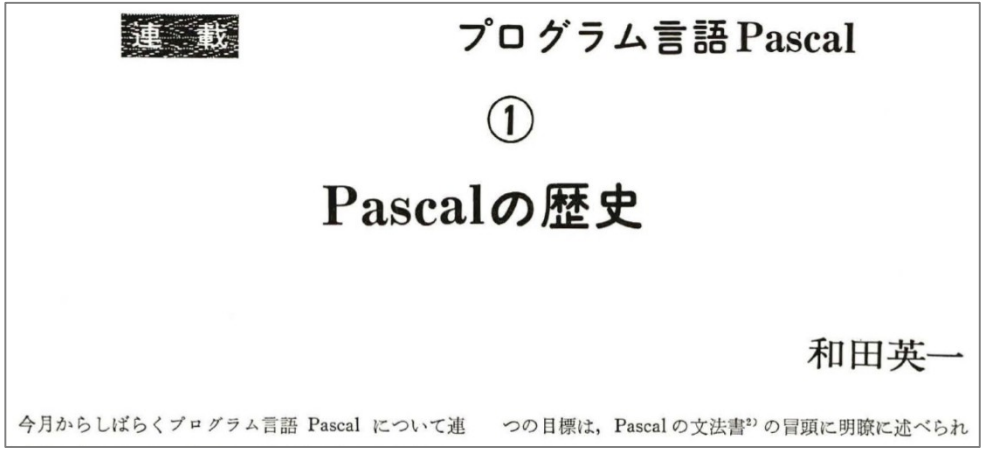
近山隆（東大名誉教授）
戸村哲（元産総研）
中島秀之（札幌市立大学長）

上田, …

- 1979年 和田先生から
  「東北新幹線小山試験線の試乗に行かないか？」
  - 試乗会（3名）の961形車内で廣瀬健先生と出会う
    （個室寝台，走行中の乗務員室等を一緒に見学）

和田研での生活が始まる

- プログラミング言語とシステムプログラム
- 1970年代は Pascal 幕開け時代
  - 学部３年授業も Pascal（サブセット処理系全容）と Lisp（後藤英一先生，他学部聴講）
- **bit** が CACM と並ぶ重要情報源


連載

プログラム言語Pascal

①

Pascalの歴史

和田英一

今月からしばらくプログラム言語 Pascal について連　つの目標は，Pascal の文法書[2] の冒頭に明瞭に述べられ

bit 1978年1月号

和田研での生活，中頃

- 先輩は言語処理系構築
  - Lisp (in Pascal)
  - Prolog (in Lisp)

bit 1978年10月号

- 言語仕様（一次資料）の精読・詳細検討
  - Pascal（← Algol W)
  - Algol 60, Algol 68, Algol N
  - Ada (US DoD, July 1980)
  - ... その他なんでも

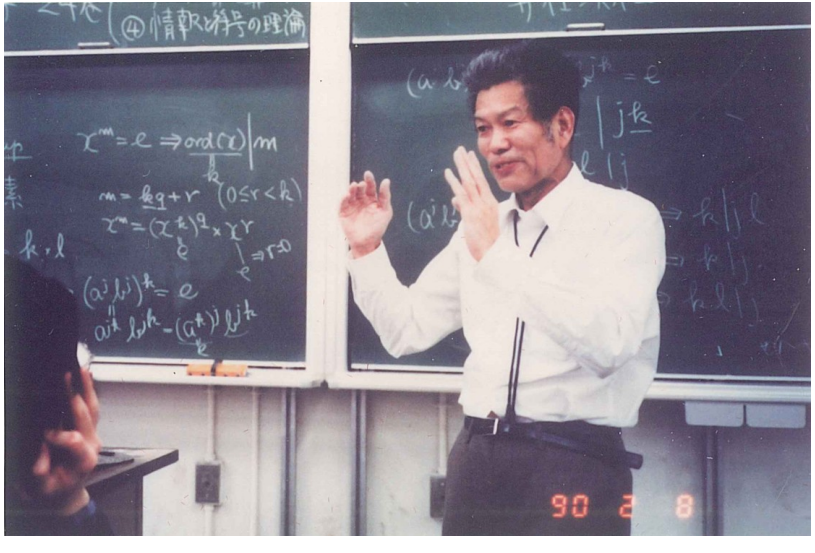
cf. 法解釈，文芸評論

和田研での生活，中頃

- 言語仕様（一次資料）の精読・詳細検討
  - Pascal（← Algol W）
  - Algol 60, Algol 68, Algol N
  - Ada (US DoD, July 1980)
    - 米田信夫先生が
      Tokyo Study Group 主宰
    - **bit** 別冊に詳説執筆
  - … その他なんでも

  cf. 法解釈，文芸評論

（C/Unix が出回り始める）



和田研での生活，終り頃

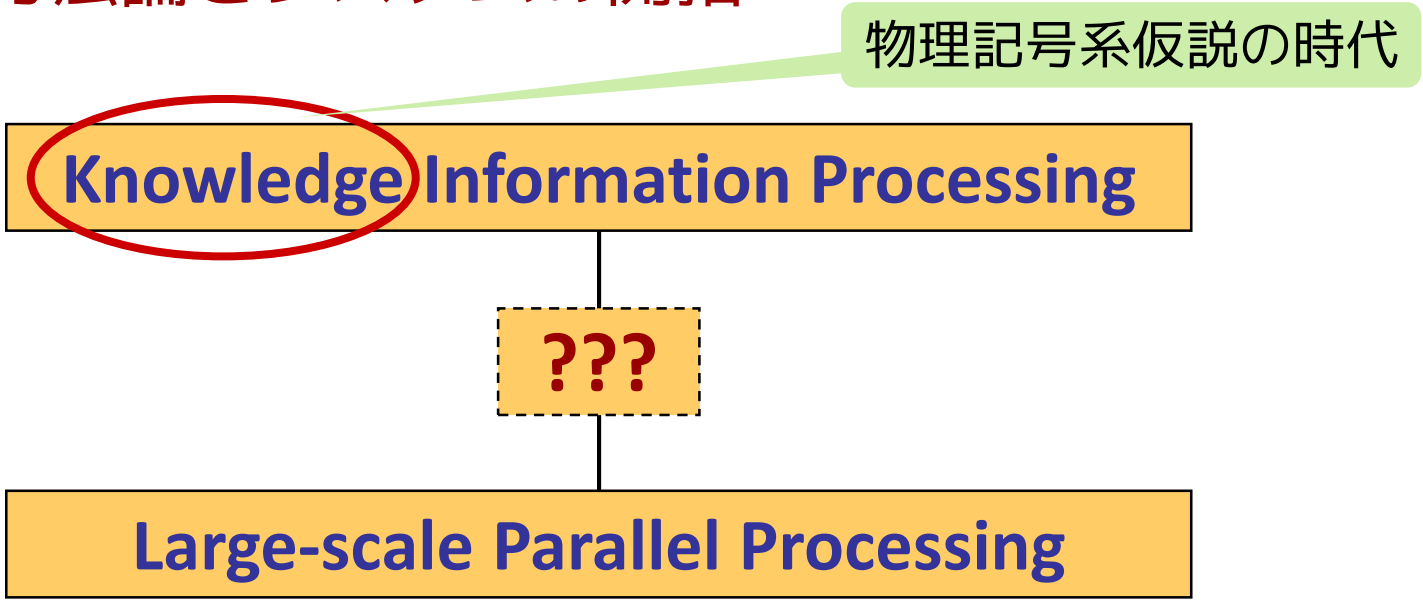
- 手続き型の太った言語は卒業したくなった
  - 中島さんの誘い：
    「PrologのI/Oをどうにかしよう」(→情処論文誌)
- 第五世代プロジェクト (FGCS) 構想が本格化
  - 官： 通産省，電総研
  - 学： 元岡達グループ 定例懇談会
  - 「プログラミング言語を作る」に飛びつく
- 1982年度末： 即決就活
  - NEC中研見学懇談，本社面接，完了

就職～早稲田着任まで (1983-1993)

NEC C&Cシステム研究所
（途中7⅓年間新世代コンピュータ技術開発機構出向）
— 第五世代プロジェクトと共に

第五世代プロジェクト (1982–1993)

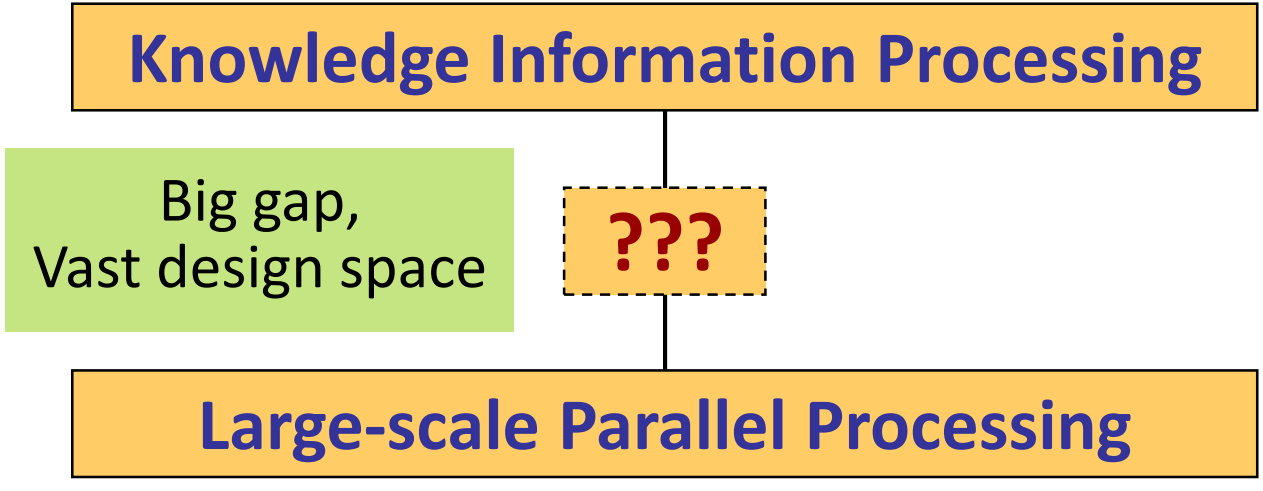
- ICOT研究員40～100名＋再委託，540億円
- **チャレンジ目標:** 知識情報処理と並列処理の架け橋となる方法論とシステムの開拓



cf. Computer architecture as the hardware/software interface

第五世代プロジェクト (1982–1993)

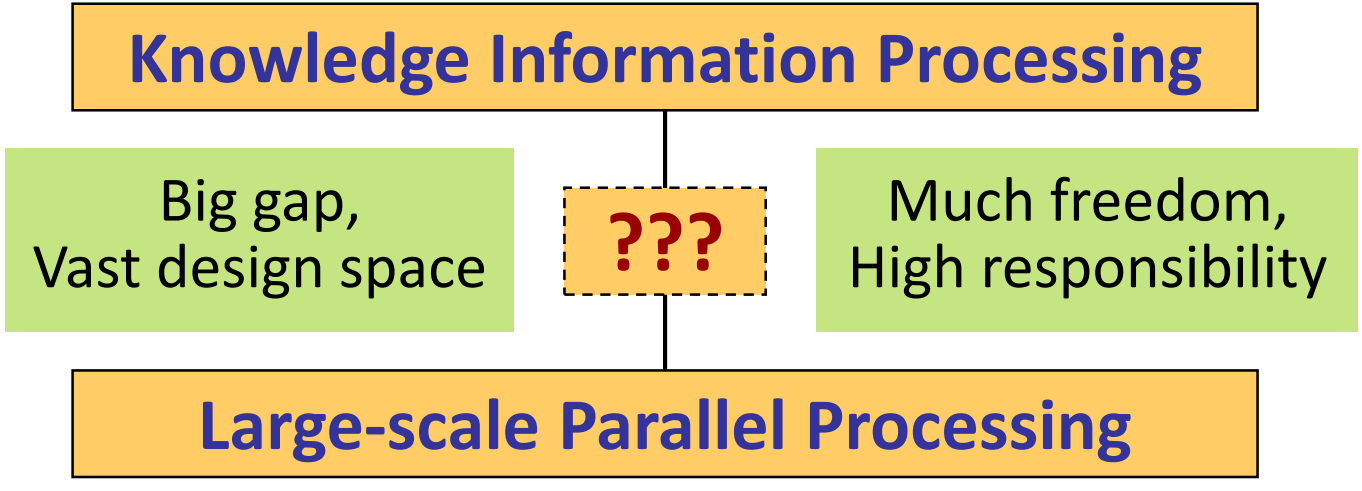
- ICOT研究員40～100名＋再委託，540億円
- **チャレンジ目標:** 知識情報処理と並列処理の架け橋となる方法論とシステムの開拓



cf. Computer architecture as the hardware/software interface

第五世代プロジェクト (1982–1993)

- ICOT研究員40～100名＋再委託，540億円
- **チャレンジ目標:** 知識情報処理と並列処理の架け橋となる方法論とシステムの開拓

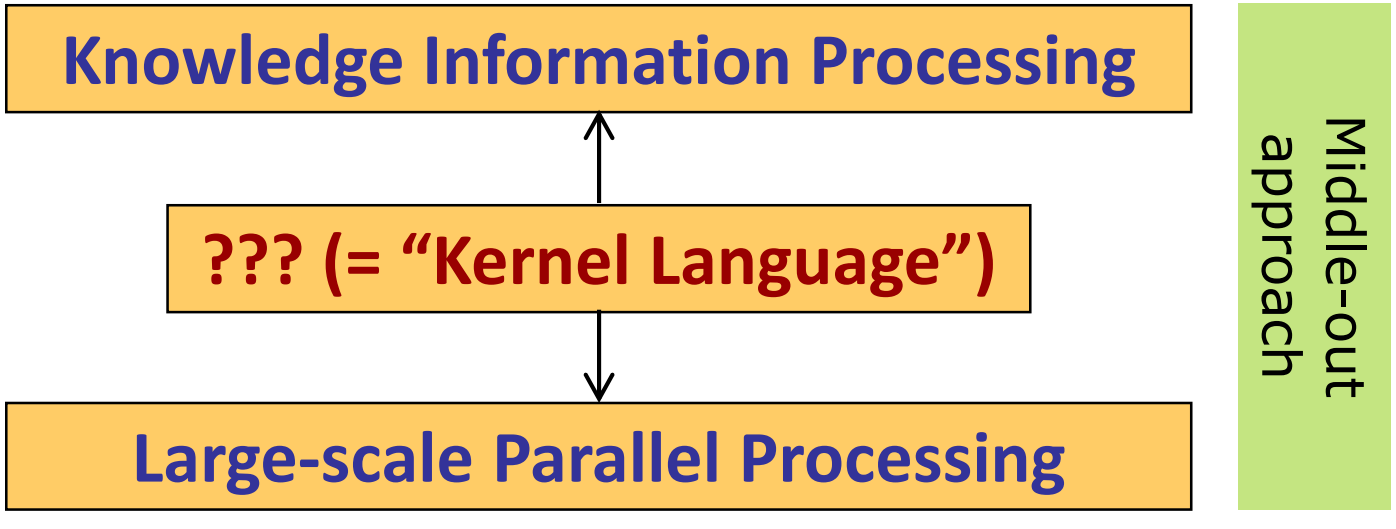


cf. Computer architecture as the hardware/software interface

第五世代プロジェクト (1982–1993)

- **チャレンジ目標:** 知識情報処理と並列処理の架け橋となる方法論とシステムの開拓
  - ➔ **（第五世代）核言語**の設計と実装が中心テーマに

AIプロジェクトではなかったの？



だれがどこで推進したか

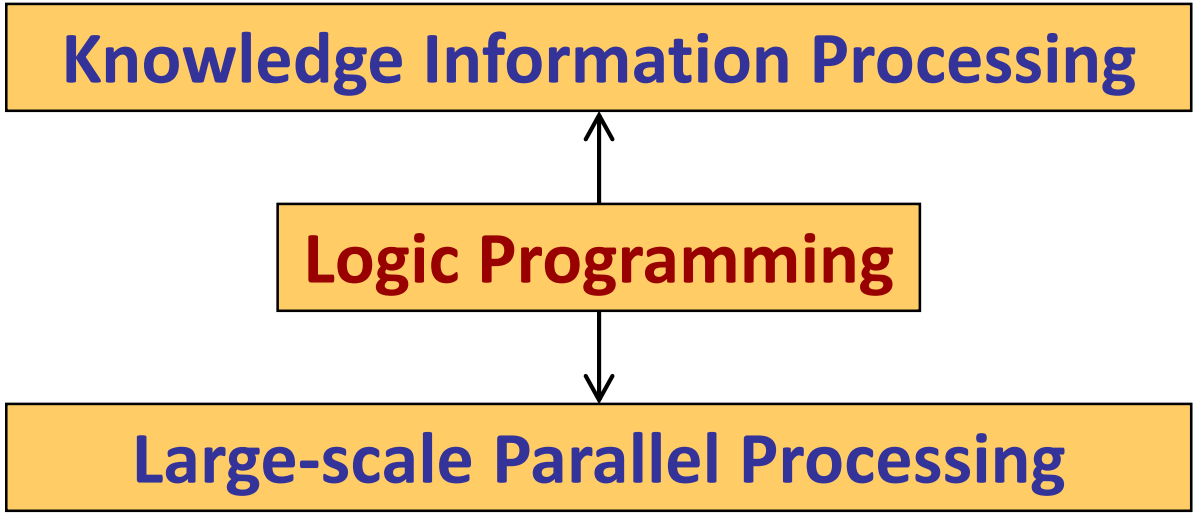
ICOT研究所 21F (+ 計算機室等)

ICOT同窓会 (2014年1月)

作業仮説 (Working Hypothesis)

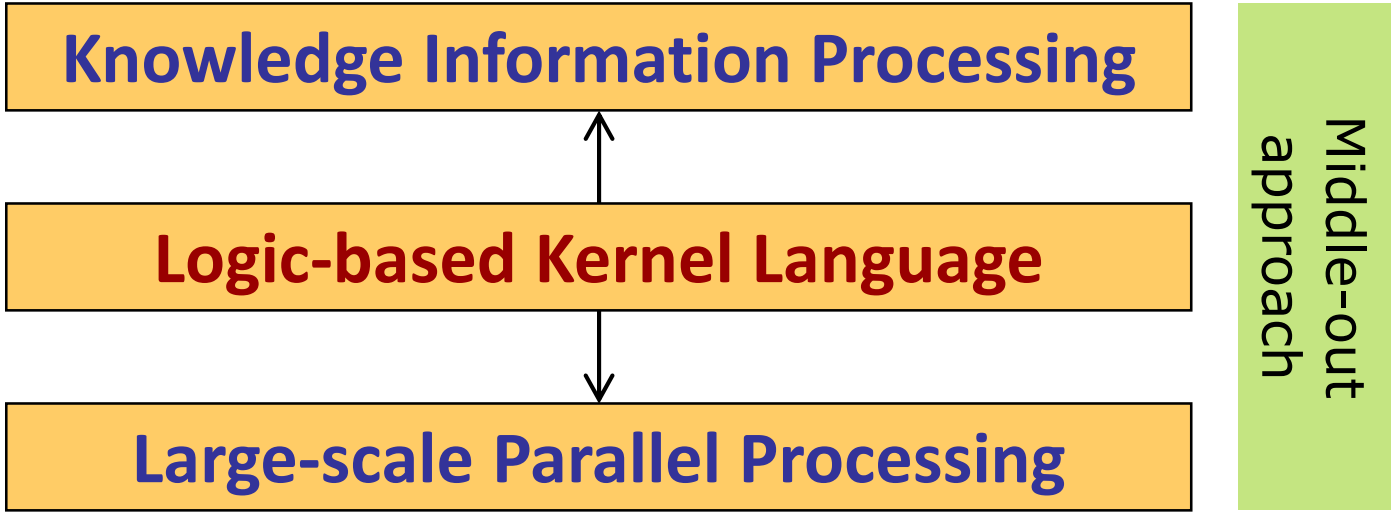
- 広い意味での「論理プログラミング」が選ばれた
  - 理由は二つ

Research questions here



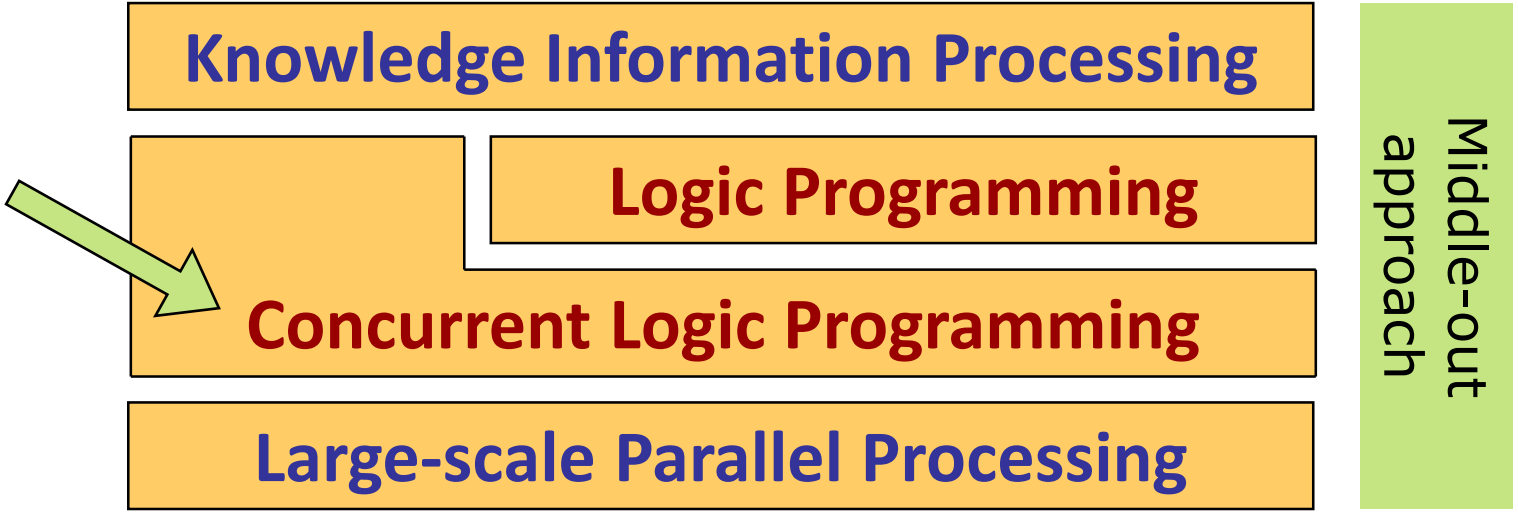
第五世代核言語 (kernel language)

- 言語づくりはプロジェクトの核であった
  - **KL0**: Sequential kernel language for startup
  - **KL1**: Parallel kernel language for systems and apps
  - (KL2: Knowledge representation language)



プロジェクト終了時の形 (1993)

- いくつかの成果
  - Kernel Language (GHC/KL1)
  - Parallel OS (PIMOS) totally written in KL1
  - Parallel Inference Machine (PIM with 512 & 256PEs)
  - Model Generation Theorem Prover (MGTP)



推論マシンの開発と利用

渕 一博 (1936-2006)

PSI-I (Dec. 25, 1983)
35KLIPS for KL0
80MB, 100 copies
SIMPOS in ESP
+ many apps

D.H.D. Warren
(WAM designer)

AITEC-ICOT
Archives
DVD, 2005

PSI-II (Dec. 1986)
330 (400) KLIPS for KL0
with 6.67MHz, WAM, 300 copies

推論マシンの開発と利用

渕 一博 (1936-2006)

PSI-I (Dec. 25, 1983)
35KLIPS for KL0
80MB, 100 copies
SIMPOS in ESP
+ many apps

D.H.D. Warren
(WAM designer)

AITEC-ICOT
Archives
DVD, 2005

PSI-II (Dec. 1986)
330 (400) KLIPS for KL0
with 6.67MHz, WAM, 300 copies

推論マシンの開発と利用

内田俊一

Multi-PSI (Mar. 1988)
64 PSI-II's in 2D mesh
5MLIPS for KL1, 6copies
PIMOS (standalone,
multi-user OS in KL1,
44KLOC in 0.5years)

PIM/m (1992)
256 PEs in 2D mesh (256 PEs)
200 MLIPS, PIMOS (200KLOC) in KL1

AITEC-ICOT Archives DVD, 2005

核言語の形成過程 [CACM March 1993 issue]

[Sci. Comput. Program. 2018]

KL1設計タスクグループ (1983-04〜)

- 古川康一（リーダー）
- 設計要件 (cf. Prolog)
  - 汎用言語
  - 並列アルゴリズムの記述
  - OSの記述
  - メタプログラミング

Keith Clark　　Ehud Shapiro

AITEC-ICOT Archives DVD, 2005

- 並行論理プログラミングの研究者を招聘して日夜議論 (1983-10)
  - Keith Clark, Steve Gregory (PARLOG)
  - Ehud Shapiro (Concurrent Prolog)

なぜ論理型言語だったか（所内討議スライド, 1984）

5Gプロジェクトの目標

「1990年代における真に汎用なコンピュータの開発」

- 数値計算ではなく 記号計算
- データ処理ではなく 知識処理
- 超並列アーキテクチャによる高速化
  （非ノイマン・コンピュータ）
- 新しい ソフトウェア科学・工学の追求

Prolog

= Lisp − α + β

+ small Relational DB

α : Functional Argument

β : Imcomplete Data Structure

$\beta >> \alpha$

並行論理プログラミング (1980～)

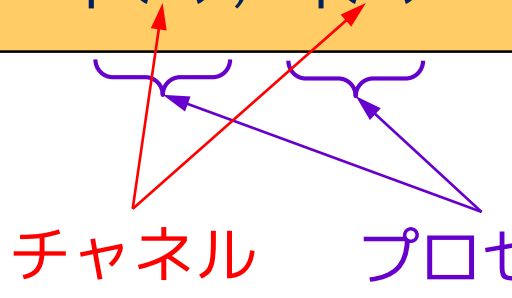
- 構文: 条件付き書換え規則 (cf. Guarded Commands)
- 意味 (cf. Prolog) :
  - データフローに基づく同期（＝計算の半順序）
    ＋ 非決定的選択 (don’t-care nondeterminism)

```
p(ok) :- true | ... .
q(Z)  :- true | Z=ok.

:- p(X), q(X).
```

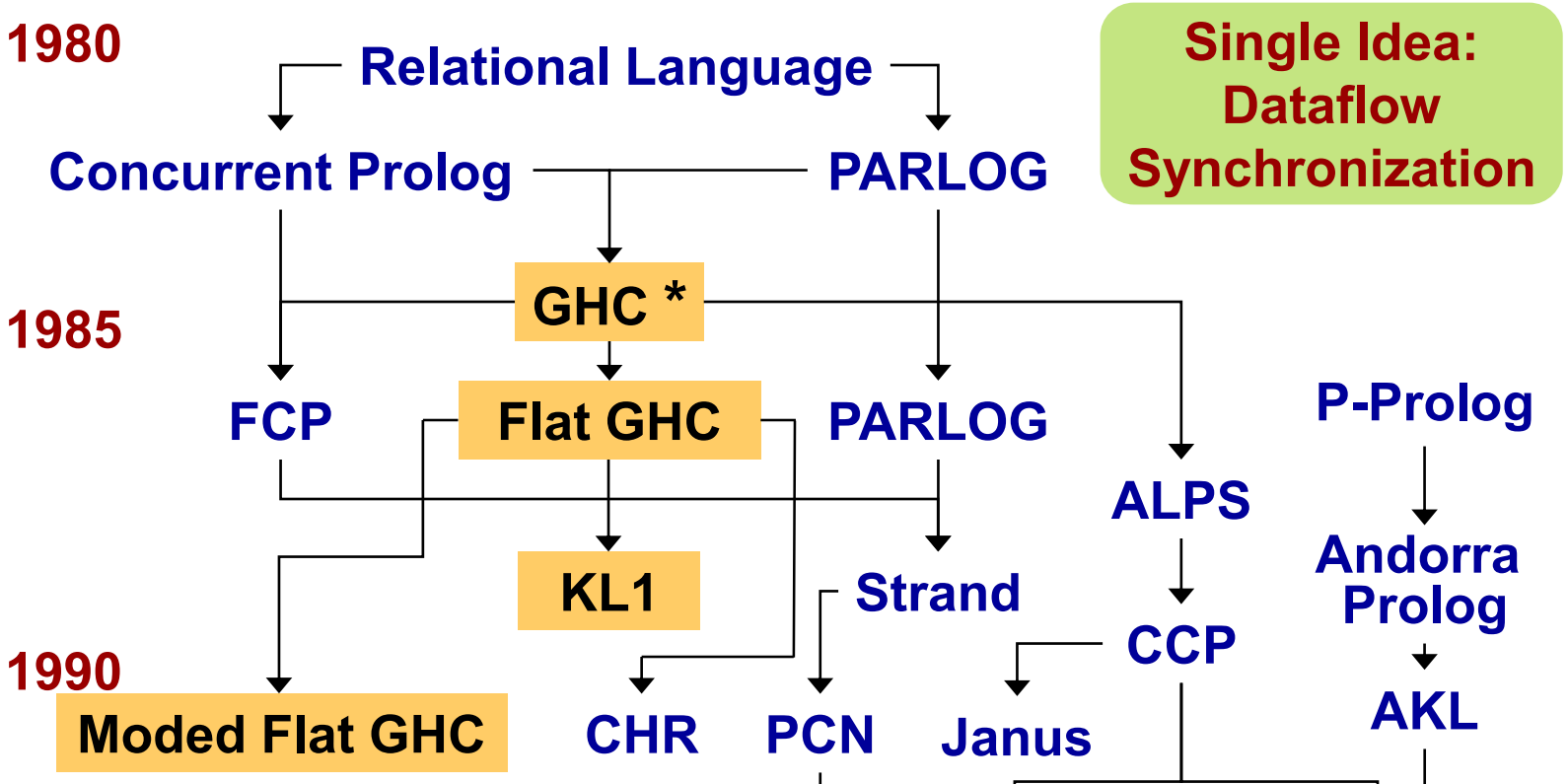
（受信, ask）

（送信, tell）



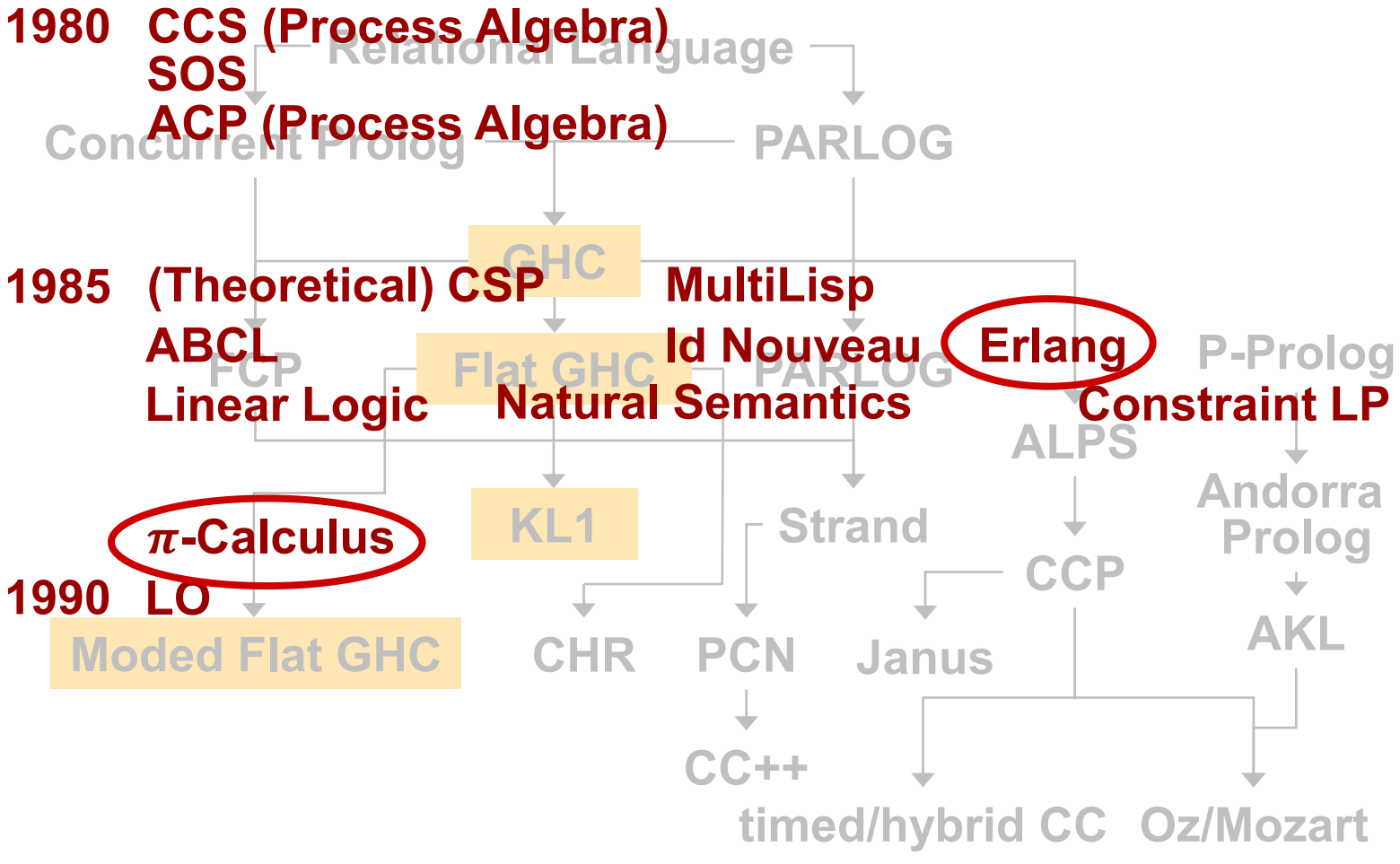
生成，消滅，変化，分裂，合体，送受信などをルールにしたがって自律的に行う実体

並行論理・制約プログラミングの初期の歴史



* REMARK: Not to be confused with Glasgow Haskell Compiler (1992)

他の計算モデルと重ね合わせると …



当時の議論・雑談から (1983)

- 鈴木則久（当時東京大学）
  “Building the entire system without resort to side effects is the FGCS project’s right way to go.”
- 内田俊一 (ICOT):
  “The computational model of KL1 should not assume any particular granularity of underlying parallel hardware.”
  (= Kernel Language should embrace as fine-grained concurrency as possible.)

当時の議論・雑談から (1983)

- Ehud Shapiro (Weizmann Institute)
  （KL1要求仕様案を見て）*“Too many good features.”*

➔ タスクグループの research question が定まる

“What’s the minimum set of language constructs that turn Logic Programming into an expressive concurrent language?” (Occam’s Razor, cf. CSP/Occam)

当時の議論・雑談から (1983)

- Ehud Shapiro (Weizmann Institute)
  （KL1要求仕様案を見て）*“Too many good features.”*

  ➔ タスクグループの research question が定まる

> *“What’s the minimum set of language constructs that turn Logic Programming into an expressive concurrent language?”* (Occam’s Razor, cf. CSP/Occam)

- 1984年2月には，Concurrent Prolog が KL1 設計のための（詳細化された）作業仮説となる
  - π 計算の意味での動的プロセス（チャネル）構造が５年以上前に実現されていた

Research question への解答

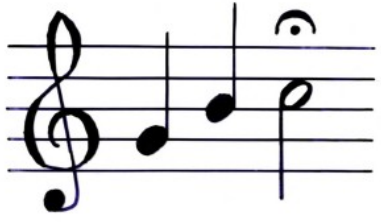
- 先輩言語の解答： 新概念の**導入**
  - 入出力モード概念 (PARLOG)
  - 変数の capability 概念 (Concurrent Prolog)
- **Guarded Horn Clauses (GHC)** (Dec. 1984) [LNCS 221]:
  - 既存概念（単一化によるデータフロー）の**制限**
  - 新たな構文要素なしに通信チャネルの動的張替えや返信箱つきメッセージを実現
  - ハードウェアグループも歓迎，KL1の新たな作業仮説に採用（1985年初頭）
  - Lesson: *simplicity was the key to the consensus.*

Guarded Horn Clauses [LNCS 221]

- プロトタイプ処理系 (DEC-10 Prolog) は１日半で作成
  - Prolog 上の Concurrent Prolog コンパイラを翻案 [SLP 1985]（まだ動きます！ 公開中）
    www.uedalab.jp/~ueda/software/ghcsystem-swi.tgz
  - Gerard Huet (INRIA) は CAML で数日で記述 (1988)
- ワイツマン研究所（イスラエル）出張，2週間議論
  - Flat GHC サブセットに移行 (1985年夏)
- 論理型言語の探索機能 (proof search) の実現法を提案 [ICLP 1986]

当時のOHPスライド (1986頃)

GUARDED HORN CLAUSES



KAZUNORI UEDA

(ICOT)

Semantics of GHC

H_1 :- $\cdots$ G_{1j} $\cdots$ | $\cdots$ B_{1j} $\cdots$.

H_2 :- $\cdots$ G_{2j} $\cdots$ | $\cdots$ B_{2j} $\cdots$.

Can be tried in //, but cannot instantiate* the caller.

Can instantiate the caller, but only one selected clause is allowed to do so.

Selects one of the clauses whose guard has succeeded (commitment)

(The other clauses cannot contribute to successful computation.)

* Instantiate: making more specific

p(X, Y) → p(1, Y) instantiated

p(X, Y) → p(A, B) not instantiated

p(X, Y) → p(A, A) instantiated

当時のOHPスライド (1986頃)

DESIGN PRINCIPLES OF GHC

1. Parallelism

- It must be a parallel language 'by nature', not a sequential language augmented with constructs for parallelism,
  - — to have a clearer semantics, and
  - — to disallow inessential sequentiality to creep in.
- Introduction of sequentiality is considered as an optimization to meet the current computer architectures.
- We have to allow even possibly useless computation.

2. Generality

- It must be a general-purpose language which can express important concepts in parallel programming.
- It must be general also in that no specific implementation scheme is assumed a priori.

3. Simplicity

- It must be a simple language because of the shortage of our experience both in the theoretical and the practical aspects of parallel programming (languages).

4. Efficiency

- It must be an efficient language which allows fast execution of simple programs at least under the current computer architectures.
  - — cf. generality (2.)
- Sequential implementation is more than a prototype.
- Efficiency may interfere with generality and simplicity, but a general language could be subsetted for more efficient execution of a specific class of programs.

リクエストはがき（当時の論文入手法）

Philippe Codognet
(→ パリ大 → フランス大使館 → … → 現・東大)

Philippe CODOGNET
THOMSON-CSF
Laboratoire Central de Recherches
Domaine de Corbeville
91 - ORSAY
FRANCE

91 ORSAY ESSONNE -8 8 86

RÉPUBLIQUE FRANÇAISE 0430 POSTES NA 42453

K. Ueda

AVION

ICOT Research Center
Mita Kokusai Bldg. 21F,
1-4-28 Mita, Minato-ku
TOKYO 108
Japon

61.8.11

M
Dear Sir : Ueda

Auriez-vous l'obligeance de me faire parvenir un tiré à part de votre article :

Please send a ~~reprint~~ preprint of the article entitled :

Guarded Horn Clauses (Tech. Report TR-103)

publié dans : / *which has appeared / will appear in :*

and also "Concurrent Prolog re-examined" (TR-10

Avec mes remerciements sincères,
Your courtesy in sending this publication to me will be greatly appreciated.

Bien à Vous,
Yours very truly,

GHC から KL1 へ

- (Flat) GHC は並行言語
- KL1 は（OSが書ける）並列言語 (PIMOS, 200KLOC)
  - プロセスからプロセッサへのマッピング
  - プロセスの保護

- Lesson: *separation of concerns*
  - 並行処理　vs.　並列処理
  - 並行処理　vs.　探索 (cf. multiparadigm languages)
  - 簡約　vs.　通信
    （「通信の不可分操作」論争, cf. π-calculus)

西早稲田で再会

古川康一
(～2017)

竹内彰一

Ken Kahn

Ehud Shapiro:
“We were 40 years ahead of time.”

2015-07-16
明治通り「串衛」

制約に基づく並行計算

— 1980年代中頃の成果を振り返ると

制約概念（≒ 等式・不等式）に基づく並行計算

- 理論：**Concurrent Constraint Programming** (late 1980's)
  - 制約論理プログラミングに触発された
    並行論理プログラミングの定式化 (Vijay Saraswat)
    - 通信と論理が結びつき，形式意味論が確立
  - Xerox PARC に招かれ Ken Kahn, V. Saraswat らと
    2 週間の言語設計論戦 (1989)
- 実践：二つの構成要素が実用の鍵
  - **単一代入チャネル** (write-once channel)
  - **コンストラクタ**（データ構造作成手段）

(cf. CCS, CSP, π, etc.)

単一代入チャネル

- 論理変数の応用 (logical / single assignment / immutable)
- 書き込みは１回だけ
  - チャネルの値についての部分情報（制約）を ***tell*** (publish) する
    - e.g., `tell S=[read(X)|S']`
  - rest of communication
- 読み出しは非破壊的
  - ある部分情報が帰結 (entail) できかを ***ask*** する (*matching*)
    - e.g., `ask ∃A ∃S'(S=[A|S'])`
  - π 計算の *input* と *match* にほぼ相当

制約概念に基づく通信機構

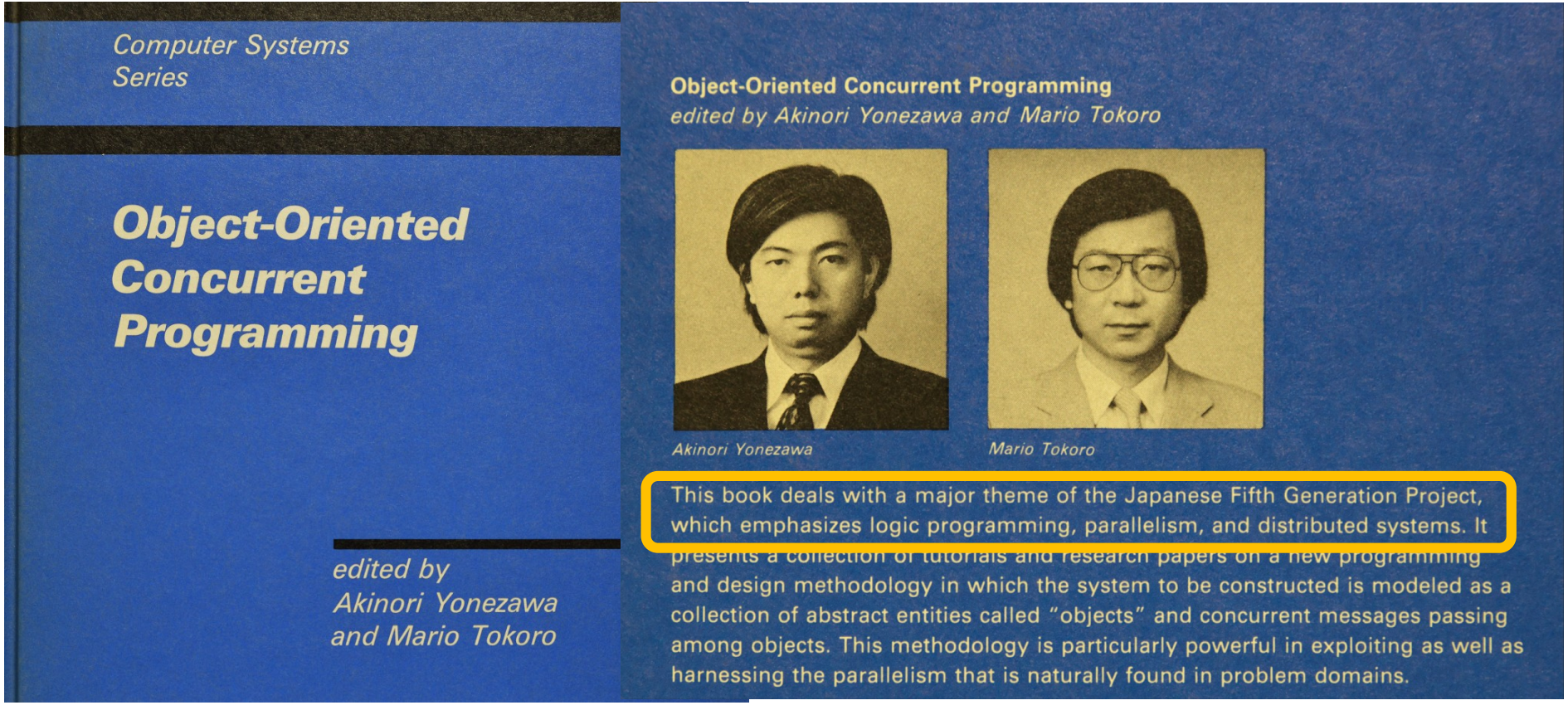
- 単一代入チャネルによる通信は以下の機能を実現した (cf. π-calculus)
  - チャネルを使ってチャネルを送る
  - 返信箱つきメッセージを送る
  - チャネルを相互結合する (fusion)
- データ構造の non-strictness を最大限利用
  - 制約ベース = *computing with partial information*
- 共有変数は使うが，本質は，当事者間のみが知る局所変数を使った peer-to-peer の通信

1980年代の計算モデル研究 ～日本ソフトウェア科学会草創期

- 多様な並行計算モデルと言語の開花
  - Actors　Carl Hewitt (1944-2022)
  - 並行オブジェクト (Concurrent Smalltalk, ABCL, …)
  - 並行論理型・並行制約　Robin Milner (1934-2010)
  - プロセス代数 (CCS, CSP, π計算, …)　Tony Hoare (1934-2026)
- オブジェクト指向の outbreak
- 第五世代コンピュータプロジェクト
  - 「並列知識情報処理」のモデルと実証実験
- 多数の外国人研究者の来日 (Hewitt@Keio, ’89-’90, etc.)

1980年代の計算モデル研究

- 並行オブジェクト指向計算は良きパートナーだった



並行論理プログラミングから生まれたもの

- **Concurrent Constraint Programming** (late 1980's)
  - 制約論理プログラミングに inspire された一般化
  - 通信機構の論理的解釈 (**Ask** / **Tell**)
  - データ領域の（有限木以外への）一般化
- **CHR (Constraint Handling Rules)** (early 1990's)
  - ゴールの多重集合の書換え系へ拡張
  - 多くの新たな応用（制約ソルバ等）
- **Timed / Hybrid CCP** (early-mid 1990's)
  - 時間，デフォルト，連続変化概念の導入
  - 時間／ハイブリッドシステムの高水準言語へ

並行論理プログラミングから生まれたもの

- **高性能並列計算と Grid のための言語** (early 1990's -)
  - PCN, CC++, HPC++, swift-lang

from Ian Foster @ ANL

> Dear Ueda-san:
> The wonders of Google Scholar citation alerts led me to your recent paper on FGCS, which I enjoyed reading.
> …
> While PCN and CC++ are long gone, we continue to work with Swift (swift-lang.org), which is really CLP in another guise.
> My best wishes from Chicago.

- **X10** (mid 2000's)
  - IBM's solution to HPC languages

第五世代プロジェクトの文献（上田）

- Logic/Constraint Programming and Concurrency: The Hard-Won Lessons of the Fifth Generation Computer Project.
  Science of Computer Programming, **164** (2018), pp.3-17
- The Fifth Generation Project: Personal Perspectives.
  CACM, **36**(3) (1993) (D.H.D. Warren & E. Shapiro, eds.)
  - 渕一博，Robert Kowalski，古川康一，上田和紀，
    Ken Kahn，近山隆，Evan Tick が個人の立場から寄稿
- New Generation Computingの草創期
  人工知能，Vol.37, No.3 (2022), pp.347-350

第五世代プロジェクトの文献（一般）

- AITEC・ICOTアーカイブズ：わが国の先端情報技術開発 (2005)
  http://www.uedalab.jp/AITEC_ICOT_ARCHIVES/
- 『第五世代コンピュータの並列処理』
  瀧和男編，bit別冊，共立出版，1993.
  http://www.uedalab.jp/~ueda/documents/（編著者・共立出版了解済）
- 渕一博記念コロキウム『論理と推論技術：四半世紀の展開』
  (2007)
  http://www.uedalab.jp/fuchi-colloquium/
- 『渕一博：その人とコンピュータサイエンス』
  近代科学社，2010.
  - 林晋：情報技術の思想家 渕一博（第１章）
- 特集：『第五世代コンピュータと人工知能の未来』
  人工知能，Vol.29, No.2, 2014.

早稲田大学 着任から現在まで

情報学科の設立準備（1980年代末）

第　1　回

早稲田大学情報科学シンポジウム

主　催：早稲田大学理工学部情報学科設立準備会
　　早稲田大学情報科学研究教育センター

協　賛：株式会社日立製作所
　　日本アイ・ビー・エム株式会社
　　日本電気株式会社
　　富士通株式会社
　　（順不同）

開催日時：平成元年６月３０日（金）　１３：００～１７：００

開催場所：早稲田大学小野記念講堂
　　新宿区西早稲田１－６－１　本部キャンパス７号館１階北側

情報学科の設立準備（1980年代末）

○オープニング　　13：00～13：15
　平　山　　博（早稲田大学　理工学部長）

○新しいオブジェクト指向言語について　　13：15～14：00
　上　村　　務（日本アイ・ビー・エム株式会社東京基礎研究所）

○論理プログラミングと並列プログラミング　　14：00～14：45
　上　田　和　紀（新世代コンピュータ技術開発機構）

（休　　憩）

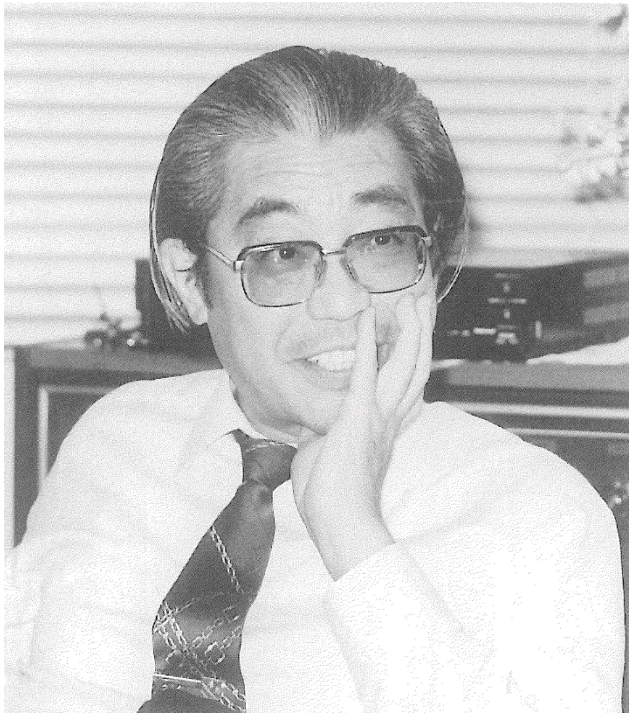
○並列処理のパラダイムを求めて　　15：00～15：45
　米　澤　明　憲（東京工業大学）

○並列処理の未来へ向けて　　15：45～16：15
　村　岡　洋　一（早稲田大学）

○認知科学のこれから　　16：15～17：00
　安　西　祐一郎（慶応大学）

情報系学部学科の設立ブーム

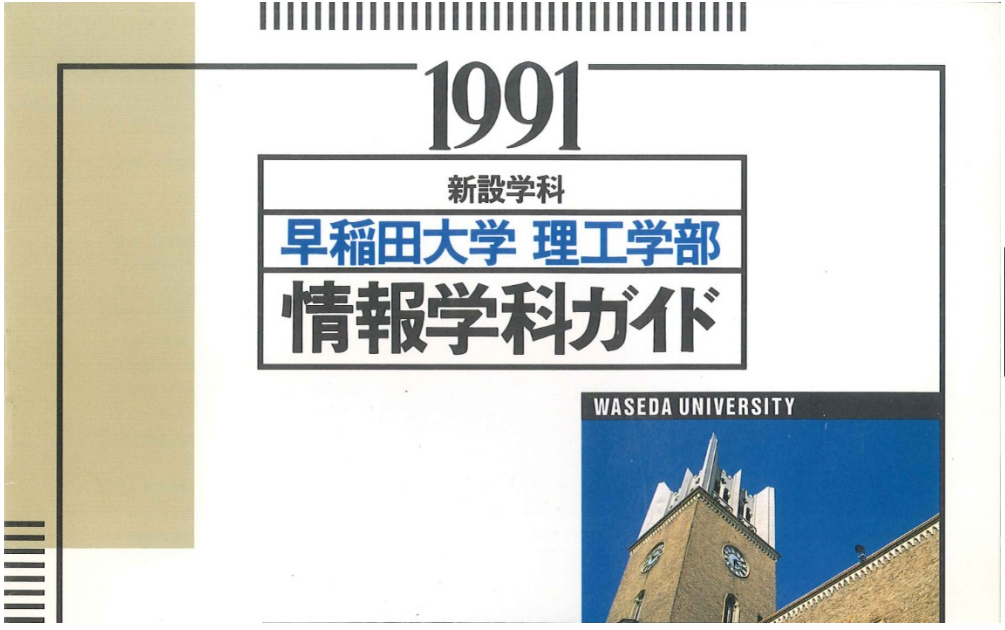
- 1980年代末～1990年代初頭
- 第五世代プロジェクト（年限付）の終了が近づき，動き出したのは研究者よりも各大学
- シンポジウムの日 (1989)，廣瀬健先生（と数名の先生方）に誘われて２回目の対面
  - 1979年の小山試験線試乗会からちょうど10年後†

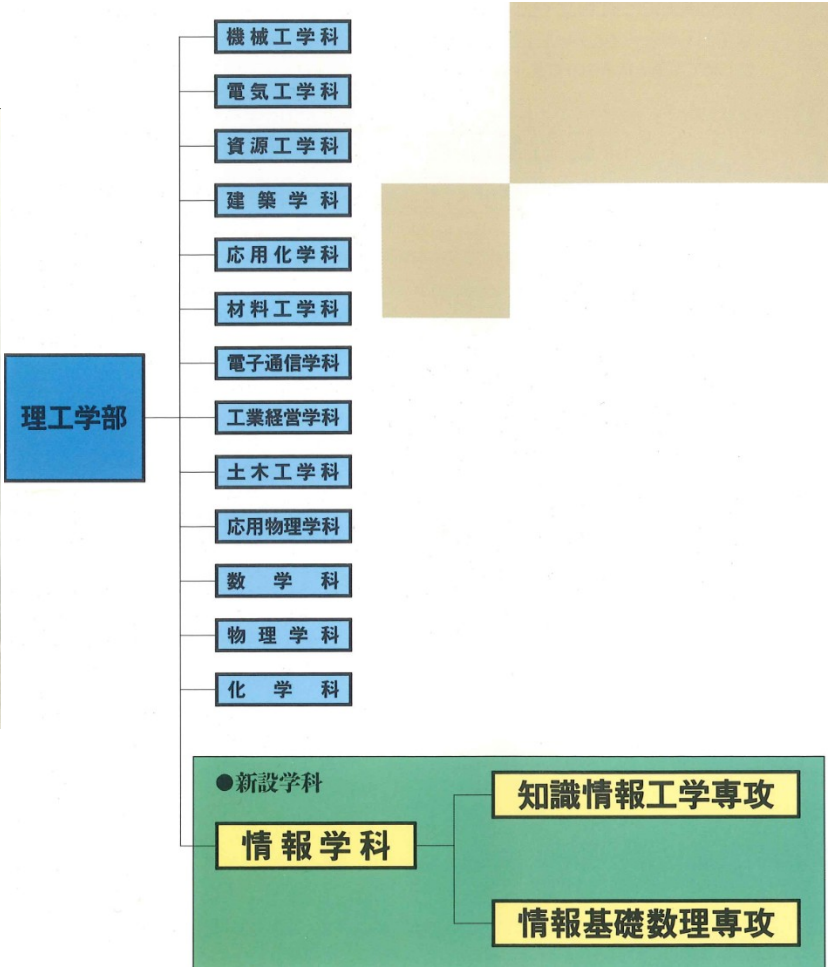


「廣瀬健の思い出」より
（亀井哲次郎氏許諾）

† その22年後 (2011-02)，大附辰夫先生から「新型はやぶさ号試乗会に行ってこない？」

設置認可 (1990-12-21)，１期生受け入れ (1991)





学科設立の経緯は
「早稲田大学理工学部百年誌」所収

情報学科に合流 (1993)

- 新設学科の何もない研究室
  - 教授も先輩もいない
  - ワークステーションは高くて買えない（リース）
  - 什器は前職プロジェクトの廃棄資産
  - 空調もない（記録的冷夏に救われる）
  - ３年生10名とゼロから立ち上げ
    - 「何もないから上田研に来た」学生たちと
    - 環境，運営方針，研究テーマ，予算，etc.
- 「並列知識情報処理研究」
- 廣瀬先生入院 (1993-06)

当時の現在地

2005-03-04

2001年ごろまでの研究開発

- 並列処理全般
  - PCクラスタ構築，アプリは検索エンジン等
  - KLIC (KL1-to-C translator)（次頁）の拡張と応用
- 並行言語の型概念の定式化と応用
  - 通信の型（⊆ データの型）(cf. session types)
    → 線形型 → 所有権と共有を扱う型 [TACS2021]
  - プログラム自動デバッガ [JSSST論文賞]（現役）
- 制約概念に基づく描画ツール
  - ハイブリッドシステム研究につながる (HydLa)

KLIC (KL1-to-C translator)

- 第五世代後継プロジェクト (1993-1994, 28億円) で開発
- 今世紀初頭まで上田研で保守,
  その後しばらく放置していたが ...
- 汎用メニーコアマシンの急速な普及（10,000円/コア）
  - KLIC復活作戦実施 [FLOPS 2016]
  - 今のシングルコア実行は，30年前の10コア実行よりも10倍以上高速
- Lesson: *Old software is lightweight and fast.*
  Why not keep it alive?

2002年ごろ以降の研究開発

- 新たなプログラミング言語の着想・設計・実装
  - グラフ書換え言語とグラフ書換えモデル検査
  - ハイブリッドシステム記述言語
  - ルーツはいずれも並行／論理／制約プログラミング
  - プログラミングとモデリングの境界の開拓
- 並列処理研究は徐々に高性能検証 (high-performance verification) にフォーカス
  - 並列SATソルバ (c-sat etc.) [SAT 2009]
  - 並列モデル検査器 (SLIM etc.)

言語をつくる

一部はJSSST2018招待講演から

技術的なことは FLOPS2026（つくば，5月下旬）の招待講演でお話しします

なぜ言語をつくるのか

- （ソフトウェアでもハードウェアでも）つくることは（論文読みよりも）楽しい
- つくるためのものをつくることはもっと楽しい
  - メタ・再帰・自己参照への興味執着はプログラミング・計算理論・数学基礎論研究者の特徴
- アート，科学，工学 が支えあう接点で仕事ができる
  - デザイン，アルゴリズム，定理，数値（性能等）のすべてを相手にできる

アート

- 工芸・技術を含む用語．芸術 (fine arts) より広い
  - “State of the *Art* ”
- Knuth: “The *Art* of Computer Programming” (1968)
  - Knuth のこだわり（チューリング賞記念講演）

Computer Programming as an Art

by Donald E. Knuth

When *Communications of the ACM* began publication in 1959, the members of ACM's Editorial Board made the following remark as they described the purposes of ACM's periodicals [2]: “If computer programming is to become an important part of computer

that is classified as an “art”; it has to be a Science before it has any real stature. On the other hand, I have been working for more than 12 years on a series of books called “The *Art* of Computer Programming.” People frequently ask me why I picked such a title; and

Commun. ACM,1974

どんな言語をつくる（つくった）のか

- 次のような言語を作ってきた
  - 非手続き型（宣言型）言語
  - 並行プログラミング言語
  - 状態空間探索と検証のための言語
- 既存の言語とは明確に異なる言語
  - “A language that doesn’t affect the way you think about programming is not worth knowing.”
    – Alan Perlis (1922-1990)，第1回チューリング賞
- すべて minimalist アプローチ
  - とは言っても処理系は数万～10万行規模以上

or making

なぜつくるのか

計算とプログラムの本質を理解したい

- ラムダ計算 (1930's) ですら相当に分かっていない
  - "We don't understand substitutions."
    – expert reviewer of RTA2008
- 既存の枠組みは，根本的な部分から疑い，考え直す価値がある

なぜつくるのか

計算とプログラムの本質を構成的に理解したい

- 設計から実装までゼロから構築すれば，生半可な理解にとどまる可能性が減るであろう
- 自由な発想で展開できる
- プログラミング言語は動いてナンボ
- 日本はものづくりの国
  - “things” + “acts of making” + “Kaizen”

どうやってつくるのか

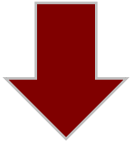
- 料理に例えると…
  - （料）素材はほとんどあり合わせ
    - 数学や論理学からとってくる
    - 既存のプログラミング言語からとってくる
    - それらの良い性質を活かす
  - （理）新たな使い方，組み合わせ方を開拓
    - “computational interpretation”
  - 「足し算」よりも「引き算」
    - 足し算 ➔ 機能間相互作用を伴うかも

懸念

- 使ってくれる人はいるの？
  - 答１: キラーアプリが出る可能性がある
  - 答２: コンセプトと proof of concept がしっかりしていれば相当に長持ちする
  - 答３: 言語の gene pool の維持発展は重要

これまでに作ってきた言語

- Guarded Horn Clauses (1984~)
  - 制約に基づく並行計算
    (constraint-based concurrency; a.k.a. CCP)
  - 関連する言語: CHR, X10, LMNtal, …



- LMNtal (2002~)
  - グラフ書換え言語とその（並列）モデル検査器
- HydLa (2008~)
  - ハイブリッド制約言語と処理系

グラフ書換え言語への展開 (2002)

- KL1の経験：同じことを書くのに二通りの方法がある
  - プロセス構造とデータ構造（cf. 関数と関係）
  - Research question: 一本化できないか？

➔ データ構造をなくしても並行プログラムは書ける

  - プロセスの集まりを書き換えてゆく言語へ
  - 単一代入変数は無代入(?) 変数へ退化

➔ グラフ書換え言語と見なせることに気づく

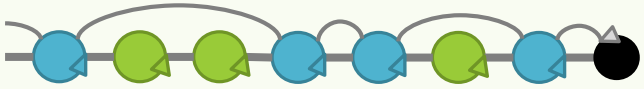
  - **L**inks, (first-class) **M**ultisets, **N**odes

http://www.uedalab.jp/lmntal/, https://github.com/lmntal

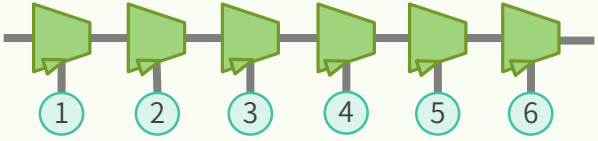
グラフを簡潔安全に扱いたい

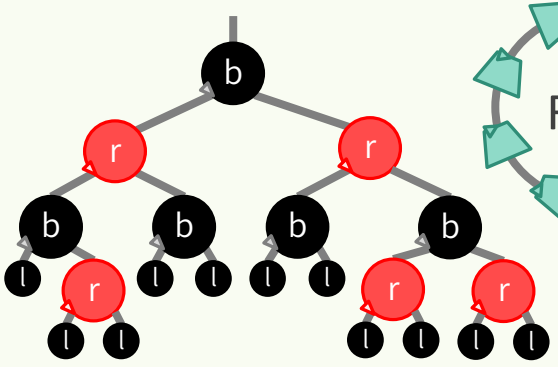
General Graph Structures

Skip list†

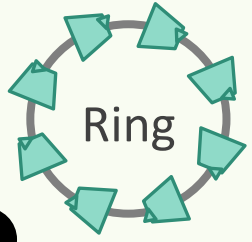


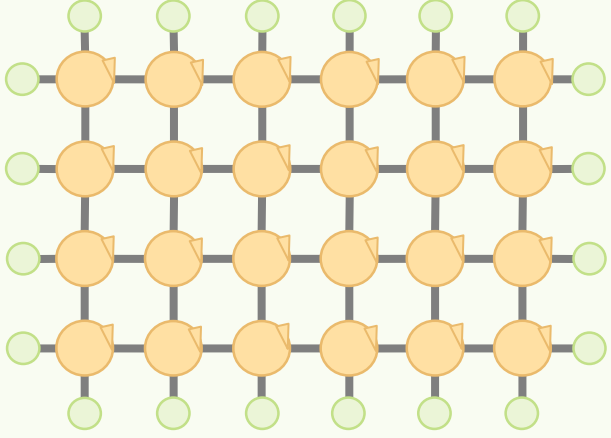
Difference list
(d-list)



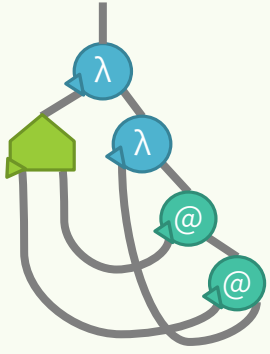


Balanced red-black tree



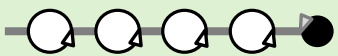


Grid

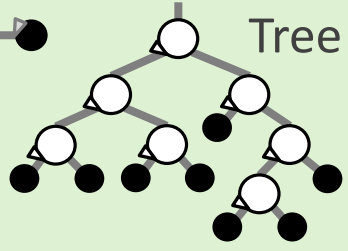


Lambda term
$\lambda f x.\, f\,(f\,x)$

Algebraic Data Types



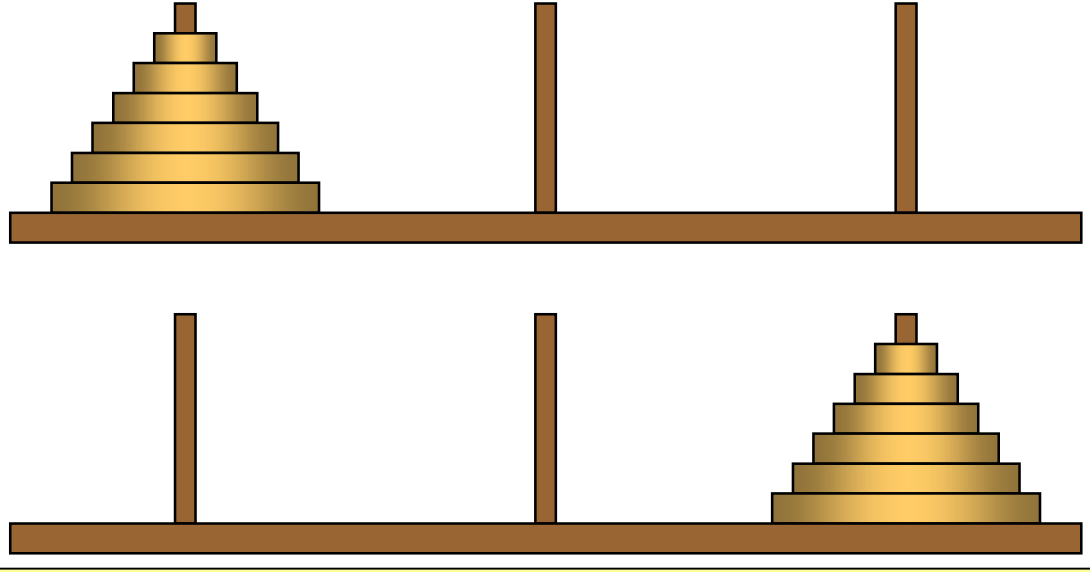
Linear list



† W. Pugh: Skip lists: A probabilistic alternative to balanced trees, Comm. ACM, 33(6), 1990.

LMNtal でのハノイの塔


```
poles(p([1,2,3,4,5,6,99]),p([99]),p([99])).
```



```
P1=p([$h1|$t1]), P2=p([$h2|$t2]) :- $h1<$t2 |
    P1=p(T1), P2=p([$h1,$h2|$t2]).
```

グラフ書換え言語への展開

- 多様な計算体系のエンコードを実現**（当初の想定外）**
  - λ計算だけで３種類作成（細粒度版x2・粗粒度版）
- 並列LTLモデル検査器の形で状態空間探索を実現
  （当初の想定外）
  - グラフ同型性判定を伴う（任せておくとできる）
  - 20億状態までスケール　（同上）
- 多対多の書換え＝書換え規則の両辺を入れ替えても構文的に正しいプログラム（対称性，可逆性）
  - 応用例：構文生成器と構文解析器

グラフ（書換え）への関心の拡大

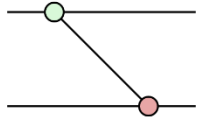
Home Tutorial Publications The ZX Seminar Accessibility Map PyZX Demo

The ZX-calculus

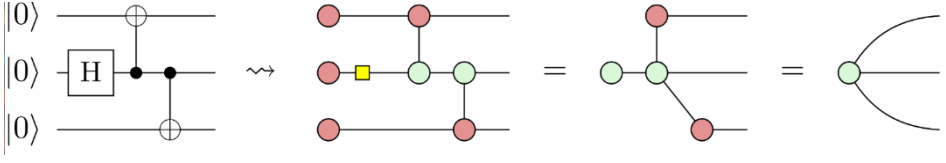
量子計算

The ZX-calculus is a graphical language that goes beyond circuit diagrams. It 'splits the atom' of well-known quantum logic gates to reveal the compositional structure inside. The calculus works by generalising the ideas of Z and X operations, allowing us to break out of the circuit model while maintaining soundness of reasoning. In doing so we can show properties of circuits, entanglement states, and protocols, in a visually succinct but logically complete manner.

The ZX-calculus is forging the next generation of quantum software. Using the calculus gives optimisation strategies that performs state-of-the-art T-count reduction (an important metric for fault-tolerant computing) and gate compilation. The generators of the calculus correspond closely to the basic operations of lattice surgery in the surface code, giving a visual design and verification language for these codes; and ZX has also been used to discover novel error correction procedures. It comes with a scalable notation capable of representing repeated structures at arbitrary qubit scales. The calculus also acts in the crucial role of an intermediate representation in a new commercial quantum compiler.



https://zxcalculus.com/



String Diagram Rewrite Theory I: Rewriting with Frobenius Structure

FILIPPO BONCHI and FABIO GADDUCCI, University of Pisa
ALEKS KISSINGER, University of Oxford
PAWEL SOBOCINSKI, Tallinn University of Tech
FABIO ZANASI, University College London

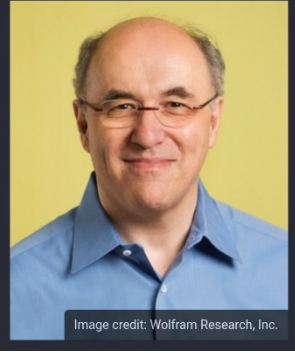
圏論

Transformation Theory and Applications Home Seminars How to participate Team GReTA-ExACT

Graph Rewriting as a Foundation for Science and Technology (and the Universe)

Stephen Wolfram

Follow



世の中全部

ハイブリッド制約言語への展開 (2008)

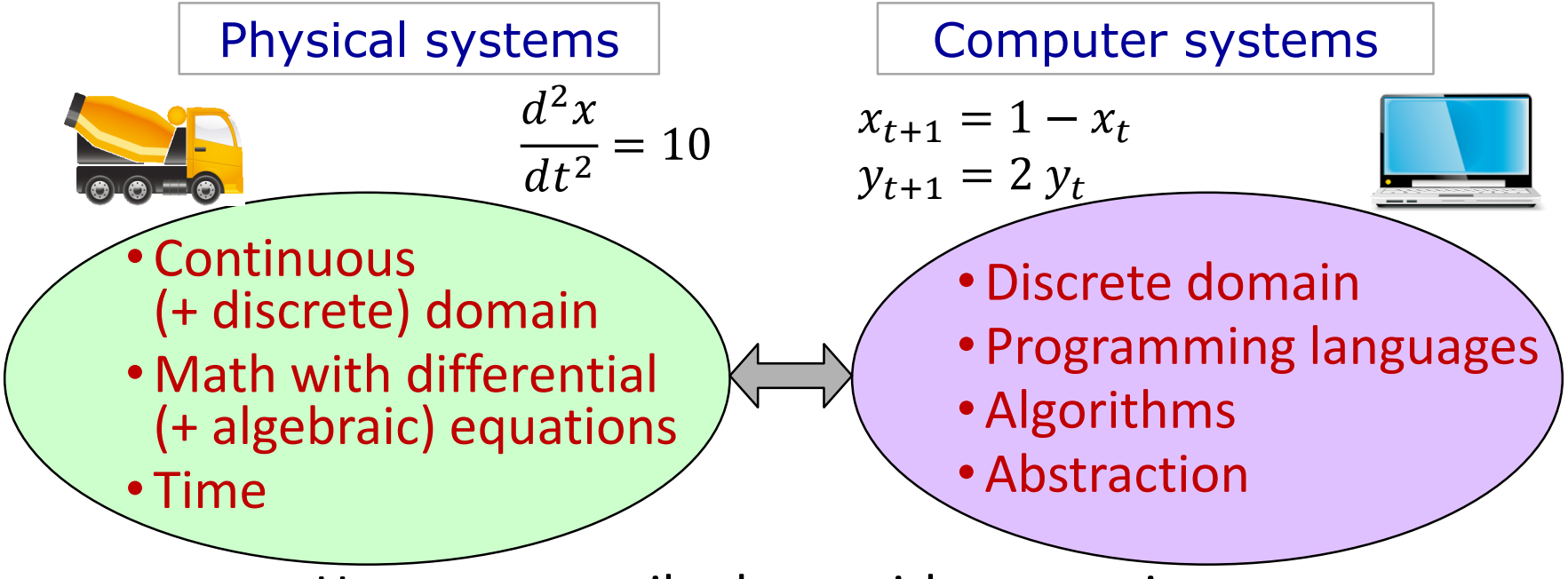
- ハイブリッドシステムは離散システムや連続システムよりも複雑で難しい（∵ 両方の要素＋α）
  - 例：ハイブリッドオートマトン
  - 微分方程式＋離散変化条件 ＝ 制約概念＋状態概念？

Research Question 1: 複雑さを最小限にしたい．（情報系以外の人にも通じる）数学と論理の記法に最低限何を加えたらモデリング言語になるか？

Research Question 2: 制約処理の基本演算は無矛盾性判定．これだけでモデリング言語が構築できるか？

ハイブリッドシステムのモデリングと計算

- Key issue
  = modeling of, and interfacing with, the *physical* world



How to reconcile them with computing abstraction of physical systems?

HydLa: ハイブリッド制約モデリング言語

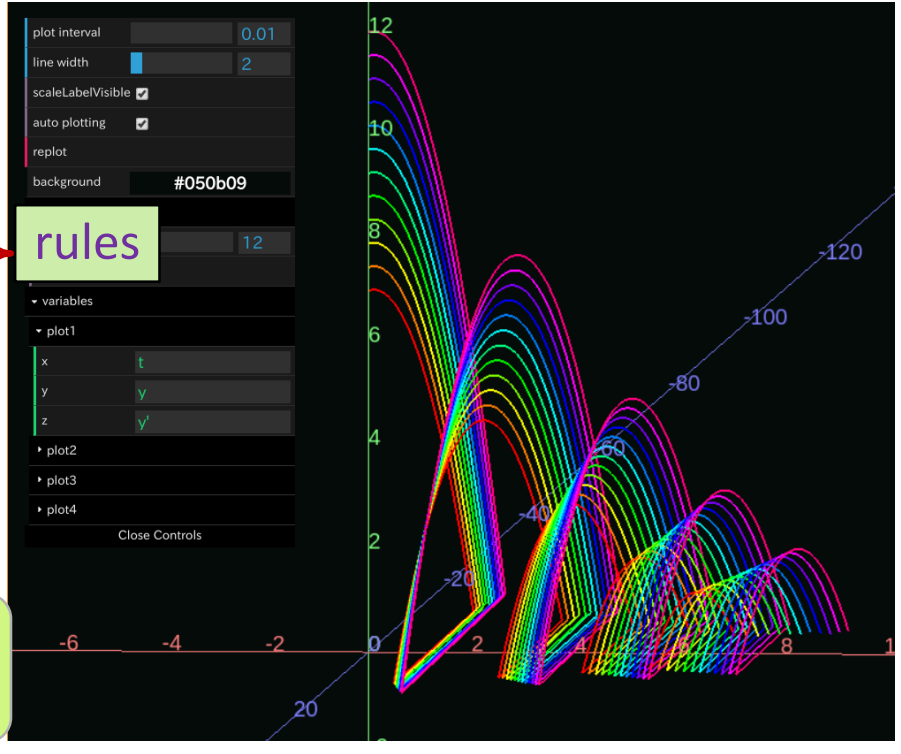
- 単なるシミュレーションとは異なる付加価値を追求
- 記号実行と精度保証（実は相性が良い）
  - 実数計算と条件判断を正しく共存させる

```
INIT      ⇔ 7 < y < 12 & y' = 0.
FALL      ⇔ □ (y'' = −10).
BOUNCE    ⇔ □ (y− = 0
                ⇒ y' = −4/5 * y'−).
INIT, (FALL << BOUNCE).
```

guard

priority

rules



弾むボールのHydLaプログラムと
webHydLa上での実行結果

研究開発の基軸は処理系の構築と発展

- 言語仕様（＋設計指針）をゼミや合宿で議論する
- 誰かが作り始める（単独 or 分担）
  - 始まったら余計な口出しはしない
  - 言語仕様への質問・フィードバックには答える
- そのうち処理系が動き始める
  - 改良のサイクルが回り始める
- たまに作り直す (e.g., Ruby → Java → C → C++)
- コードの 99% は学生の研究開発成果（cf. 海外）
  - 次に来る学生の研究基盤として自分たちで維持
  - 20年前のコードが現役稼働

ソフトウェアが幹，論文は果実

- ソフトウェア維持発展の方法：班ごとの開発合宿
  - 個別研究成果の整備と統合（稼働状態を維持）
    cf. 世の中の多くの成果は論文が出たら終わり
  - OS/コンパイラ変化への対応（怠ると動かなくなる）
  - ドキュメントの整備
  - ソースの理解（言語処理系は難しい）

  - 合宿先（ほぼ温泉地）を選べば作業は学生主導
  - CI のおかげで初級者でも安心して貢献できる

継続は力なり

LMNtal合宿	HydLa合宿
2006 湯河原温泉	2006 八王子
2007 箱根湯本温泉	2007 伊豆長岡温泉
	2008 御岳山，四季の湯温泉（埼玉）
2009 葉山（神奈川）	
	2010 御岳山（東京）
2012 秩父	2012 御岳山
2014 磐梯熱海温泉	
2015 湯河原温泉	2015 御岳山
2017 磐梯熱海温泉	2017 湯河原温泉
2018 養老渓谷温泉	2018 養老渓谷温泉
2018 湯河原温泉	
2019 石山温泉（滋賀）	2019 磐梯熱海温泉
2020 那須塩原温泉	2020 辰口温泉（石川）
…（コロナ中はリモート集中作業）…	
2023 那須塩原温泉	2023 箱根湯本温泉
2024 箱根湯本温泉	2024 養老渓谷温泉
2025 猪苗代中ノ沢温泉	2025 塩山温泉
2026 伊香保温泉	2026 八幡温泉（長野）

Software-centered 研究の良さと難しさ

- ソフトウェア vs. ソフトウェア論文
- 新たな言語を作る人が少なくなった
  - 仕様・理論・実装を皆で作り上げる過程は楽しいが…
  - 数年は論文が出せないと覚悟（若手研究者は可能？）
- ソフトウェアは保守を続けないと動かなくなる
  - アーカイブ体制（作れるはず）の未確立
- 業績や貢献評価に（対内的にも対外的にも）配慮と工夫が必要
  - 「コンピュータソフトウェア」編集委員長時代に「ソフトウェア論文」を定義して実行に移した
  - ACM には Software System Award がある

技術的なまとめ ー 最近考えていること

- 帰納的な議論がしにくい対象は研究テーマの宝庫
  - プロセス網，グラフ，ラムダ抽象，…
- 一つのプログラミング言語が実は複数の言語からなることが多い
  - 実行の記述＋型の記述＋検証仕様の記述
  - ばらばらでよいのか？
- 部分情報と記号実行も研究テーマの宝庫
  - 実行とプログラム解析の統合に向けて
  - 外延と内包の関係の見直し

そろそろ終わります

2012-04-06


ご来場ありがとうございました
これからもご厚誼のほど
よろしくお願いいたします