SADへの招待

森田　昭夫

- SADとは
- SADで何が出来る
- SADの情報
- SADを使うには

SADとは(生い立ち)

- Strategic Accelerator Designの略称
  - 1986年頃からKEKで開発されてきた加速器設計用の計算機コード
    - 名前は、CERNのMAD(Methodical Accelerator Design)への対抗
    - Ad hocかつ Purpose Drivenな開発
      - 必要とする人が必要な機能を(自分が使うために)実装してきた
      - (自分に)必要がなくなった機能を保守しない＆取り外さない
      - (自分に必要ない)ドキュメントを残さない
        - ソースコードがドキュメントだと豪語する人も…
    - Kitchen Sink Approach
      - 必要な道具はすべて取り込む
        - 機能間の連携・直交性や互換性は考慮されていない
    - 1994年頃に、本格的(チューリング完全・計算完備)なスクリプト言語(SADScript)インタプリタが実装された
      - 計算機で計算可能なものは、何でも記述できる
      - 現在では、Runtimeでの拡張や他言語の関数呼び出しをサポートしている
        - 2006年に Dynamic Shared Object Loader導入(Shared Object経由の機能拡張)
        - 2012年より FFICall拡張の同梱開始(Cの任意関数呼び出し機能)
    - CVSによる管理は、1995年から
      - それ以前の変更履歴は不明

SADとは(実態)

- 実装は、Fortran + C
  - Fortranの配列オフセット(Integer)でポインタを代用
    - 表現域がInteger型の語長に依存
    - 基準点が、配列の配置に依存
  - ISO Fortran/ISO C準拠では無いコードがある
    - 言語仕様上の動作未定義・実装依存となるコードを含んでいる
      - 動作は、コンパイラの実装に依存する(保証されない)
      - 移植は自己責任で!
- 仕様書・退行試験の不在
  - 最終製品では無い
    - 使用者は、開発者ないしα・βテスターですよね?
  - 退行や不具合の混入は日常茶飯事
    - 報告が有っても治る保証は有りませんが、報告されない(認識されない)バグは治りません!
- 改訂履歴の不備
  - CVS repositoryはあるが、commitlogに有意な情報が少ない・tagが貼られていない
    - 近年の MAIN trunkに関しては、Backport commitと taggingは管理しているが、k64-1-6 branch(oide branch)に関しては、管理が行き届いていない

SADで何が出来る(1)

- 加速器モデリング
  - 加速器ビームライン・パラメータの定義環境 (MAINレベル環境)
    - 標準的な加速器要素(DRIFT, BEND, QUAD, …)
    - 汎用な加速器要素(MAP) - プログラム可能な加速器要素(非物理的な過程も含むことが可能)
  - 4種類の計算エンジン
    - 幾何学配置計算(GEOMETRY)
    - 光学関数計算(TWISS)
    - ビームマトリクス計算(EMIT)
    - 粒子追跡(TRACK)
  - 対話的なビームラインマッチング環境 (FFSレベル環境)
- 汎用プログラミング環境
  - チューリング完全・計算完備な言語処理系 SADScriptインタプリタ (FFSレベル環境内)
  - 豊富な組み込み機能
    - 数学関数(三角関数・指数関数・Bessel関数…)
    - 文字列操作・リスト操作
    - GUI Toolkit Tkinter(Tcl/Tk widgetのWrapper)
      - 汎用のグラフ表示
      - 加速器に特化したグラフ表示
    - File/Network I/O
    - EPICS Channel Access
    - プロセス制御・無名共有メモリ
    - Dynamic Shared Object Loaderによる動的な機能拡張
      - FFICall(関数プロトタイプ経由の任意関数呼び出し)

- 主要な文法規則が3種類存在する
- プログラミング次第で何でも出来る

SADで何が出来る(2)

- 加速器ビームラインの定義を構成する要素
  - ドリフト空間 - DRIFT
  - 標準的な電磁石 - BEND, QUAD, SEXT, OCT, DECA, DODECA
  - 加速空胴 - CAVI, TCAVI
  - 電磁石･加速空胴の複合体 - MULT
  - ソレノイド磁場の境界 - SOL
  - ビームビーム相互作用 - BEAMBEAM
  - アパーチャー - APERT
  - 汎用の写像 - MAP
    - 4種の計算エンジン毎の写像をSADScriptで定義出来る
  - ビームモニタ･マッチング基準点 - MONI, MARK
  - ビームライン上の要素の並び - LINE
- 加速器ビームラインのパラメータ (MAINレベルの環境変数)
  - MASS – 粒子の固有質量(eV/c^2)
  - CHARGE – 粒子の比電荷
    - 標準値(1)以外での動作は、保証されていない(というか、異常動作する要素が存在する)
  - MOMENTUM – 粒子の基準運動量(eV/c)

SADで何が出来る(3)

- 加速器ビームラインの計算エンジン
  - GEOMETRY
    - 3次元空間上の幾何学配置を計算する
  - TWISS
    - 光学関数を計算する(4x5行列モデル)
    - ビーム進行方向に関する光学関数は扱えない
  - EMIT
    - ビームマトリクス及びその固有値を計算する(6x6行列モデル)
  - TRACK
    - 6次元の粒子追跡を行なう
    - 疑似乱数発生器を使ったシンクロトロン放射光モデルを含む
  - これら4種の計算エンジンは部分的なコード共有はあるが、大部分は独立しているため、相互の計算結果の整合性は一般には保証されない
    - TRACKの結果を統計処理して得たビームマトリクスと EMITの結果
    - EMITの結果から得られる光学関数と TWISSの結果

SADで何が出来る(4)

- 対話的ビームラインマッチング
  - 標準状態では、入力をUpper Caseへ変換して照合する
    - Lower Caseを含むビームラインエレメントは照合に失敗する
      - 詳しくは、PRSVCASE/CONVCASEフラグを参照
  - 代表的なFFSレベルの対話コマンド
    - USE – ビームラインの選択(類似 VISIT/BYE)
    - FIT - マッチング条件の設定
    - SHOW - マッチング条件の表示
    - VARY – 各要素のマッチングパラメータの設定
    - MIN･MAX･MINMAX - パラメータの可変範囲の設定
    - FREE - マッチングに参加する要素の設定
    - VAR - マッチング変数の表示
    - GO - 光学関数(TWISS)のマッチングを開始する
    - CALC – 光学関数(TWISS)の計算を行う
    - DISP - 光学関数を表示する
    - EMIT - ビームマトリクス(EMIT)の計算を行う
    - 詳細は、公式Webの SAD/FFS Command & SAD Script (通称SADHelp)を参照のこと

SADで何が出来る(5)

- SADScriptインタプリタ
  - チューリング完全・計算完備な言語処理系
    - 実行速度やデータ量の制限を除き、計算可能な処理は全て実現する能力がある
      - 計算科学的には万能計算機チューリングマシンと等価な計算能力
  - Wolfram Mathematicaを意味論のモデルとした”項書き換え処理系”
    - Mathematica完全互換では無い
    - 入力された式を構成する項を書き換え規則に基づいて置換してゆく処理系
      - 命題に公理・定理を適用し変形する操作を機械化したものに相当
      - 式や項の再帰的な構造をリスト構造として扱っている
    - 適切な書き換え規則の初期集合を与えることで、手続き型言語処理系や関数型言語処理系を模擬している
    - ユーザー定義の手続きや関数の導入は、書き換え規則へ新たな規則を追加する操作として定式化されている
      - 極論すれば、変数の定義・代入も”記号列を定数に置換する規則の追加・改変”である
    - 次の性質から関数言語よりの処理系になっている
      - 第一級オブジェクトとして、関数リテラルが存在する(Function・純関数型)
      - 主要な項書き換え規則は、リストの頭部(先頭要素)の種類に応じたリスト操作処理を行なう規則で構成されている
      - リスト操作を行なう数多くの高階関数(関数を引数とする関数)が標準で提供されている
    - 学習用の教材としては、Mathematicaでのプログラミング解説書がある程度参考になる
    - リスト処理や高階関数の考え方に関しては、LispやHaskell等の関数言語の解説書が参考になる

SADで何が出来る(6)

- SADScriptが扱えるデータ(Atom・原子)
  - Symbol
    - アルファベット・$を先頭にアルファベット・数字・$で構成される記号列
  - Real
    - IEEE754倍精度浮動小数点を内部表現とする数値データ
    - 10進数表現(指数部有り)の他に、0xを接頭辞として16進数表現をサポート
      - Ex) 1.2e-3, 0x1.fep+3
    - 有限数値とならない表現として、INF(無限大)やNaN(Not a Number)をサポート
  - String
    - 長さを持ったByte列(Unibyte String)
    - “ もしくは ' で、囲まれた記号の列
      - \によるエスケープ文字のサポートあり
        - “\n” - 改行文字
      - \###(8進数)や \x##(16進数)による Single Byte Code Pointをサポート
        - “\0” - ASCIIコード 0(NULL文字)
        - “\x0a” - ASCIIコード 0x0a(改行文字)
      - \u#### による Unicode Code Point(UTF-8エンコード)をサポート
        - “\u03b1” - U+03b1(α) バイト列としては、0xce 0xb1

SADで何が出来る(7)

- SADScriptが扱えるデータ(List･リスト)
  - 広義のリスト
    - 頭部を持つデータの並び 頭部[data1, data2, …]
      - data#は、何らかの原子もしくはリスト
    - 原子でないデータは、全て(広義の)リストである
  - 狭義のリスト
    - 頭部が Listシンボルであるリスト List[data1, data2, ...]
      - 演算子表現として、{data1, data2, …} が与えられている
  - 特殊なリスト(代表的なもの)
    - 複素数
      - Complexシンボルを頭部とした 実部と虚部の組 Complex[実部, 虚部]
      - 定数シンボル I の評価値は Complex[0, 1]
    - 純関数
      - Functionシンボルを頭部とした式
        - Function[{Symbol列}, 式] もしくは Function[式]
      - 内包する式を定義とした関数を表す
    - 複合式
      - CompoundExpressionシンボルを頭部とした式の集まり
        - CompoundExpression[式1, 式2, …]
        - 式1; 式2; … (演算子表現)
      - 内包する式を順に評価し、最後に評価した式の値を CompoundExpressionの値とする
      - 例外的ケースを除けば、SADScriptのプログラムは CompoundExpressionである

SADで何が出来る(8)

- SADScriptが扱えるデータ(その他)
  - ベクトル・行列
    - 数値型(Real・Complex)を並べたListや2重Listをベクトルや行列と同一視して扱う
      - 特に、Realのみを並べたListは特別な内部表現を持っている
  - 構造体
    - ラベルとデータのペアの集合を表現するリストとして扱う
      - ラベルとデータのペアの表現には、置換規則として扱われる特別な頭部 Rule, RuleDelayedを持つリストを割り当てる
        - Rule[label, data] or RuleDelayed[label, data]
        - label->data or label:>data
      - 全体は、RuleもしくはRuleDelayedのリスト
        - {label1->data1, label2->data2, …}
  - 演算式
    - 数値データ(数値・ベクトル・行列…)の演算処理の記述もリストである
      - Times[Plus[a, b], Plus[a, Times[-1, b]]]
        - 演算子表現では、(a + b) * (a + (-1 * b))

SADで何が出来る(9)

- SADScriptの組み込み関数(数学関数)
  - 剰余 - Mod
  - 最大･最小 – Max, Min, MinMax
  - 絶対値･符号関数 – Abs, Sign
  - 階段関数 – Floor, Ceiling, Round
  - 複素数 – Complex, Conjugate, Re, Im
  - 指数関数 – Exp, Log, Sqrt
    - SAD/Math/MatrixFunctions拡張 - MatrixExp, MatrixLog
  - 三角関数 – Sin, Cos, Tan, Sinh, Cosh, Tanh, ArcSin, ArcCos, ArcTan, ArcSinh, ArcCosh, ArcTanh
  - Bessel関数 – BesselJ, BesselY, BesselI, BesselK
    - 左半平面での漸近解との接続や、原点を回る際の解析接続に問題がある
  - ガンマ関数 – Gamma, LogGamma, GammaRegularizedQ, GammaRegularizedP
  - 誤差関数 – Erf, Erfc
  - フーリエ変換 – Fourier, InverseFourier
  - 行列演算 – Dot, Innter, Outer, Det, Transpose, LinearSolve, Eigensystem, SingularValues, IdentityMatix, DiagonalMatrix
  - 求根 – FindRoot
  - 非線形回帰 - Fit

SADで何が出来る(10)

- SADScriptの組み込み関数(文字列操作)
  - 組み込みの文字列操作は、Unibyte Stringに対するもので、Multibyte文字や Wide文字を認識しない
  - 部分文字列 – s[index], s[begin, end]
  - 長さ - StringLength
  - 検索 - StringMatchQ, StringPosition
  - 結合・削除・置換 - StringJoin, StringInsert, StringDrop, StringFill, StringReplace
  - Case変換 - ToUpperCase, ToLowerCase
  - ASCIIコード変換 - ToCharacterCode, FromCharacterCode
  - 文字列-式変換 - ToString, ToExpression, SymbolName, Symbol
    - 文字列化を制御する環境変数 - $FORM, PageWidth

SADで何が出来る(11)

- SADScriptの組み込み関数(リスト操作)
  - 生成 – Table, Range
  - 長さ - Length, Depth, Dimensions
  - 抽出 - Take, Drop, First, Rest, Last, Level
  - 結合 - Append, Prepend, Join.
  - 変換 - Reverse, Flatten, Thread, Partition
  - 集合操作 - Sort, Union, Intersection, Complement
  - 要素操作 – Part, Insert, Delete, ReplacePart, Extract, FlattenAt
  - 置換 – ReplaceAll, ReplaceRepeated
  - 構造演算子 – Apply, Map, MapThread, MapAll, MapIndexed, MapAt, Scan, ScanThread, Nest, Position, Count, Cases, DeleteCases, SelectCases, Select, SwitchCases

SADで何が出来る(12)

- SADScriptの組み込み関数(制御構造)
  - 条件分岐 – If, Which, Switch
  - ループ – Do, While, For
    - リストの個々の要素に対する操作は、Apply, Map, Scan等の構造演算子の適用が推奨される(コード量・速度的に)
  - 総和・総積 – Sum, Product
  - 中断 – Break, Continue, Return, Exit
  - 例外 – Throw, Catch

SADで何が出来る(13)

- SADScriptの組み込み関数(書き換え規則定義)
  - 規則定義･改変 - Set(=), SetDelayed(:=), UpSet(^=), UpSetDelayed(^:=)
  - 規則削除 – Unset(=.), Clear
  - SetとSetDelayedの違い
    - 式1 = 式2 – 式1を式2の定義時点での評価結果へ置換する規則の定義
      - 式1 := “式2の評価結果” と同義
    - 式1 := 式2 – 式1を式2へ置換する規則の定義
      - その場で式2の評価を行なわない故に Delayed 演算子
      - 式1がパターンを含むと、実行時に部分式が照合され、パターンに照合した部分式は、式2から参照可能
        - この場合、書き換え規則がある種の関数適用として振る舞う
  - UpSet, UpSetDelayedでは、式1での部分式照合手順が若干異なる(上方値参照となる)

SADで何が出来る(14)

- SADScriptの組み込み関数(Tkinter)
  - Widget – Window, Frame, LabelFrame, TextLabel, Button, CheckButton, RadioButton, MenuButton, Entry, SpinBox, ListBox, Scale, ScrollBar, Canvas, TextEditor, TextMessage, Menu, OptionMenu
    - Tk widgetとほぼ 1:1 に対応
  - フォント選択 – TextFont
  - イベント処理 – Update, TkSence, TkWait, TkReturn, TkCreateFileHandler, TkDeleteFileHandler

SADで何が出来る(15)

- SADScriptの組み込み関数(File/Network I/O)
  - 入出力 – Read, Write, WriteString, Seek
    - 入出力の識別子は、Fortranの論理ユニット番号を採用している
    - 77番は、起動時の入力台本に割り当てられる
  - File I/O - OpenRead, OpenWrite, OpenAppend, Close
  - Netwotk I/O - TCPOpen, UDPOpen, TCPAccept, TCPShutdown, Close
  - Network Name Resolver – GetHostAddrByName, GetHostNameByAddr
  - I/O Polling – SelectUnit, PollUnit

SADで何が出来る(16)

- SADScriptの組み込み関数(EPICS Channel Access)
  - CaOpen系 – CaOpen, CaClose, CaRead, CaWrite
    - Unix File Descriptorをモデルにした同期インターフェース
  - CaMonitor系 – CaMonitorクラス
    - ca_search_and_connectと call back handlerによって実装されたクラスベースの非同期インターフェース
  - 両者のbackend実装が異なるためレコード操作時の挙動が異なる(特にWaveformレコード)

SADで何が出来る(17)

- SADScriptの組み込み関数(プロセス制御･共有メモリ)
  - プロセス制御 – Fork, Wait, Wait4, Kill, Sleep/Pause, Execve, System
    - ID操作 – Get*ID, GetPGID, SetPGID
  - シグナル制御 – SigPending, SigSuspend, SigProcMask
  - 環境操作 – Environments, GetEnv, SetEnv, UnsetEnv, Umask, GetDirectory, SetDirectory
  - 共有メモリ – OpenShared, ReadShared, WriteShared, Close

SADで何が出来る(18)

- SADScriptの組み込み関数(Dynamic Loader/FFICall)
  - Dynamic Loader – DynamicLink, DynamicUnlink, DynamicCall
    - Shared Objectの動的結合･結合解除および Shared Object上に実装されている SADScript関数の呼び出し
    - 機能拡張のお供に
      - 自分が高速化したい処理を Native Binaryで実装する
      - 内部処理用の関数に勝手口を実装する(例. ExtendedTwiss拡張)
  - FFICall拡張 – FFICall
    - 関数シンボル名と引数プロトタイプを元にスタックフレームを生成し、目標となる関数を呼び出す(呼び出すだけなら任意の関数を呼び出せる)
    - SADScript関数を実装せずに必要な処理を呼び出す
      - libm上の数学関数を直接呼び出す(例. cbrt, hypot, fma)

SADの情報(Web Page)

- 公式 http://acc-physics.kek.jp/SAD/
  - 接続時のレスポンスが悪い事が多いので、気長に…
    - おそらく、サーバーのMac miniがスリープ状態になってる
- 公式掲示板
  http://acc-physics.kek.jp/cgi-bin/SAD2/wforum.cgi
- SAD Wiki
  http://www-kekb.kek.jp/Documentation/SAD/
- Source Code Browser
  http://afsad1.kek.jp/viewvc/viewvc.cgi/oldsad/
- Issue Tracker http://afsad1.kek.jp/redmine/

SADの情報(Manual)

- SAD Help(公式Web内)
  - 加速器モデルの定義・FFS対話環境のコマンド・一部SADScript関数の解説
    - http://acc-physics.kek.jp/SAD/SADHelp.HTML
- SAD/Tkinterの使い方(公式Webにて配布)
  - SADScript言語及びGUI Toolkitである Tkinter全般の解説書
    - 一部、旧仕様の記述がある(Tkinterの Canvas系など)
  - SADScript言語の文法(構文要素、演算子とその結合規則・順位)を含むのでコードを読み解くためには必読
    - 字句解析レベルの定義は、情報が古い(近年の拡張を含まない)ないしは記述が曖昧
      - 正確な動作を理解するには、実際の動作を検証する・ソースを確認する等が必要です
- KBFrame Manual(SAD Wiki内)
  - KEKBで UI構築に使われた TkinterのWrapper
    - 細かな挙動はスクリプトを参照(/SAD/share/Packages/KBMainFrame.n)
    - 解説されていない裏オプションも、実装に追加されている
    - http://www-kekb.kek.jp/Documentation/SAD/index.php?FFS%20Level%2FKBFrame

SADの情報(Tutorialほか)

- SAD Wiki Tutorial
  - http://www-kekb.kek.jp/Documentation/SAD/index.php?Tutorial
- SAD School 2011の講義録
  - http://accphys.kek.jp/indico/categoryDisplay.py?categId=8

SADを使うには

- SAD計算機(afsad)の利用を推奨
  - 64bit OSで動作するSAD環境を構築する難易度は高め
  - 現在、ソース配布用のミラーサーバーが不在のためソースの入手性に難あり(年度内には解決したい…)
    - SAD計算機のアカウントと CVSの知識が有れば取り出せます
      - CVS repository(/SAD/cvsroot/oldsad)を複製し、複製から checkout
        - repositoryへの書き込み権限が無いと cvsコマンドが動かないため
- 対話環境を起動する
  - /SAD/bin/gs FFS
- 台本(foo.sad)を実行させる
  - /SAD/bin/gs foo.sad

2回目以降の講師と題材は？

- プログラミング言語的なアプローチ
  - 既存のコードを読み解くための基礎
    - 字句・構文規則、基本的な予約語、名前空間、意味論
      - 主に、MAINレベル、FFSレベル、SADScriptの3種がある
  - コードを書く上での基礎
    - 関数型言語入門、上方値束縛、置換、入出力
- 加速器モデリング的なアプローチ
  - 単粒子力学、MAINレベルの加速器要素、MAINレベルの字句・文法規則、FFSレベルコマンドの使い方
- 実務寄りのアプローチ
  - モデルケースを想定したプログラミングとハンズオン

提供可能な実務的ハイエンドネタ

- DSL over SADScript
  - 字句規則･構文規則から意味解析、特にシンボルのスコープや内部表現の違いから来る挙動の解説等を起点に、リスト処理･クラス固有の名前空間を使った演算子オーバーロード等を利用したドメイン固有言語(DSL)の作成技法の紹介
- Inside of SAD Backends
  - SADのNetwork/並列制御/Tkinter/CaMonitorを支えるBackendの実装技術の解説及び、実装上の制約からくる不可解に見える動作の解説と対処法