SAD-X and PTC ?

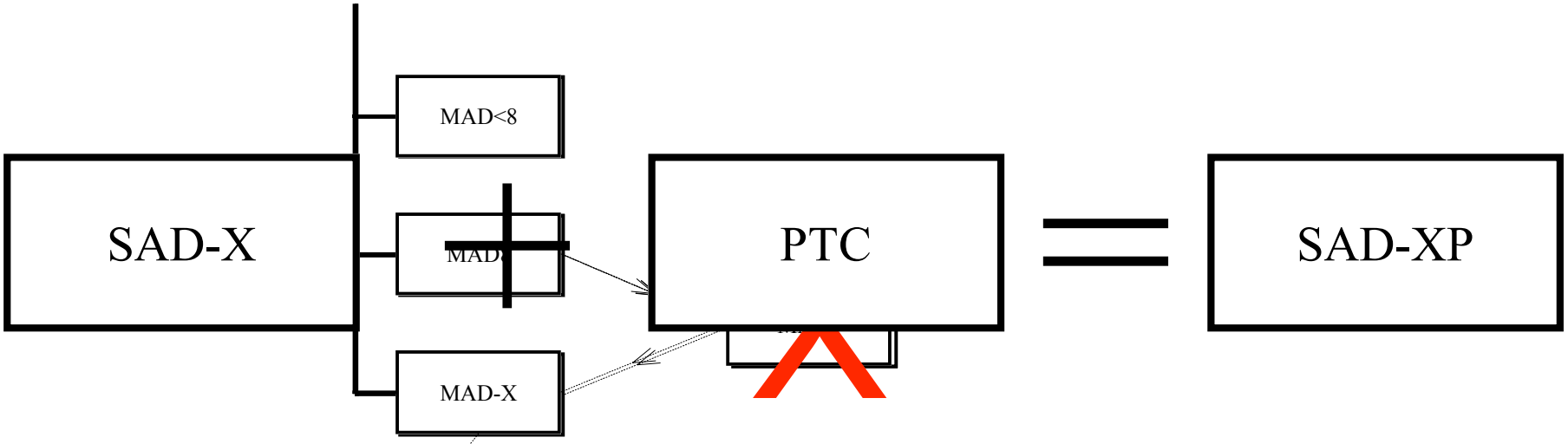
What is PTC?

フォレ　エティエンヌ

Forest　Etienne

KEK

Why no MAD-X in my Talk?



Why not? SAD is a super set of MAD
and is no worse than MAD at accepting PTC.
Furthermore a future SAD-X could internally
be set up to accept PTC!

So no need to talk about MAD in this general concepts presentation.

What is PTC?

- Contains two parts

1. FPP => Overloading of Berz and Forest FORTRAN77 tools for production and analysis of Taylor Maps. Creation of a convenient Polymorphic Taylor Type.
2. PTC proper=> A single particle tracking code with structures fully exploiting the "s"-maps of magnets which also uses FPP.

FPP

- Stands for fully polymorphic package. It is only useful if you know what to do with a Taylor map
- In PTC, I use it mainly for lattice function calculations and other perturbative calculations.
- Nothing new there if you know my work.

Example in a Tracking Environment

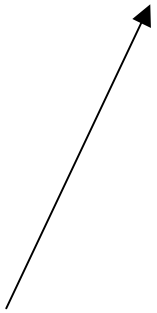
```
SUBROUTINE EXAMPLE_TWISS
USE ORBIT_PTC
IMPLICIT NONE
INTEGER I,NO
REAL(DP) BETA_X
TYPE(REAL_8) X_POL(6)
TYPE(DAMAP) ID
TYPE(NORMALFORM) NFORM
CALL ALLOC(X_POL);

X_ORBIT=0.D0; X_ORBIT(5)=0.01D0;
 CALL FIND_ORBIT(MY_RING,X_ORBIT,1)
WRITE(6,*) X_ORBIT

WRITE(6,*) &
"GIVE ORDER OF TAYLOR MAP ";
READ(5,*) NO;

IF(NO>0) THEN
 CALL INIT(NO)
 CALL ALLOC(ID);
CALL ALLOC(NFORM);
 ID=1
 X_POL=X_ORBIT+ID
ELSE
 X_POL=X_ORBIT
ENDIF

DO I=1,MY_ORBIT_LATTICE%ORBIT_N_NODE
 CALL ORBIT_TRACK_NODE(I,X_POL)
ENDDO

WRITE(6,*) "POLYMORPHS PRINTED "
CALL PRINT(X_POL,6)

! WE WILL DO A TWISS CALCULATION OF BETA_X
IF(NO>0) THEN

 NFORM=X_POL
 WRITE(6,*) NFORM%TUNE
 X_POL=X_ORBIT+NFORM%A_T
 BETA_X= (X_POL(1).SUB.'10')**2+(X_POL(1).SUB.'01')**2

  WRITE(6,*) 'POS = ',0,'       BETA_X = ', BETA_X
 DO I=1,MY_ORBIT_LATTICE%ORBIT_N_NODE
  CALL ORBIT_TRACK_NODE(I,X_POL,DEFAULT)
  BETA_X= (X_POL(1).SUB.'10')**2+(X_POL(1).SUB.'01')**2
  IF(MOD(I,50)==1.OR.I==MY_ORBIT_LATTICE%ORBIT_N_NODE) &
  WRITE(6,*) 'POS = ',I,'        BETA_X = ', BETA_X
 ENDDO

ENDIF

CALL KILL(X_POL);
IF(NO>1) THEN
 CALL KILL(ID);
ENDIF
END SUBROUTINE EXAMPLE_TWISS
```

What is structurally new in PTC?

Outside Accelerator Physics

- “Real World”: Newton-Lorentz Equation

$$\frac{d\vec{p}}{dt} = e\,\vec{v} \times \vec{B}$$



Primordial input

Describing the global B field is what matter

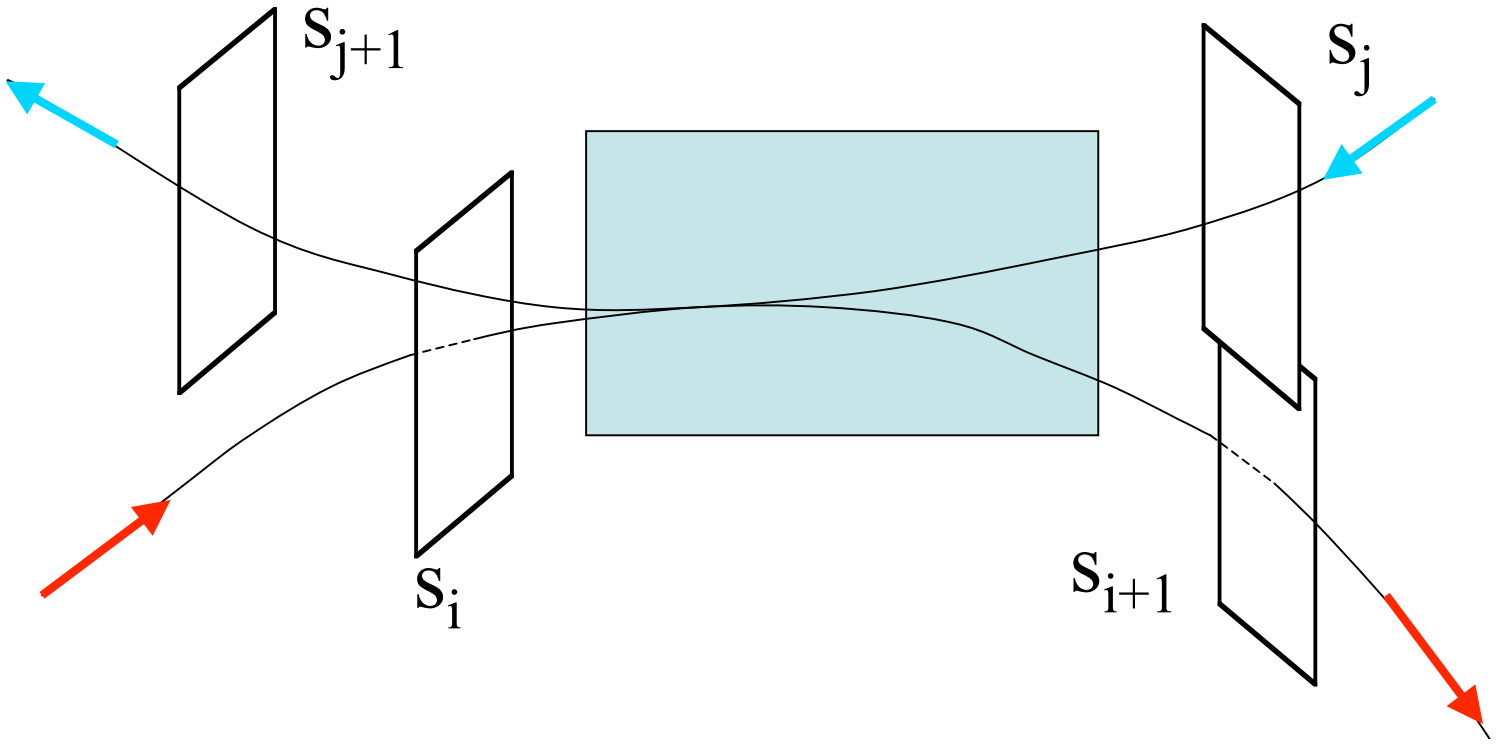
Results

- A map which, given B (or/and E), tells us the position and the velocity of a particle after a time Δt

$$(x, p_x, y, p_y, z, p_z)_{t_0} \overset{Map}{\longmapsto} (x, p_x, y, p_y, z, p_z)_{t_o+\Delta t}$$

Accelerator Code

$$(x, p_x, y, p_y, t, p_t)_{s_j} \overset{Map}{\longmapsto} (x, p_x, y, p_y, t, p_t)_{s_{j+1}}$$



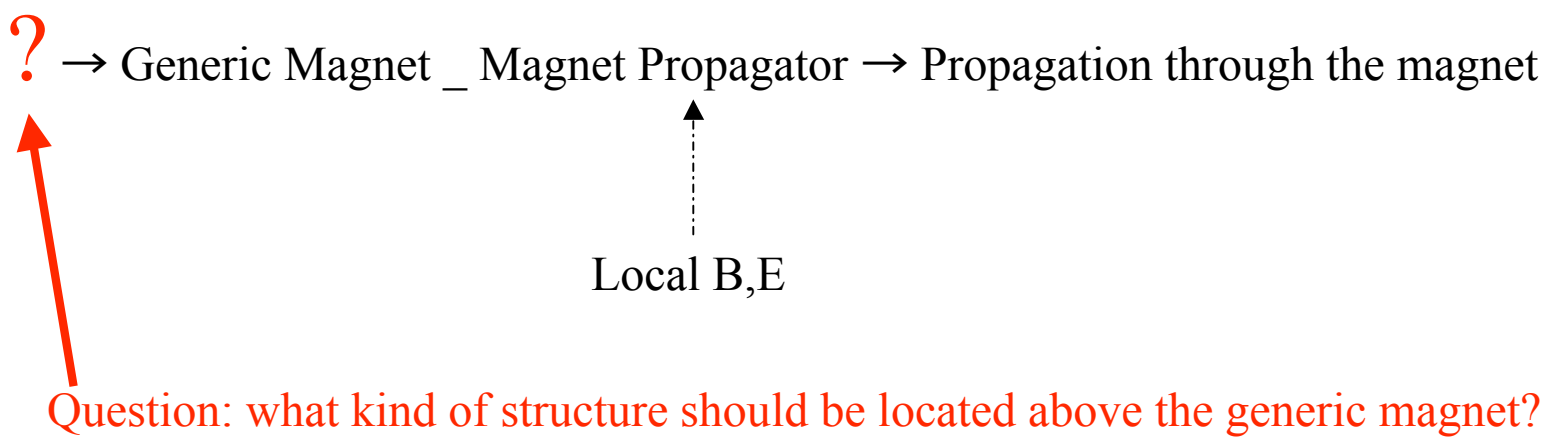
$$(x, p_x, y, p_y, t, p_t)_{s_i} \overset{Map}{\longmapsto} (x, p_x, y, p_y, t, p_t)_{s_{i+1}}$$

Summary

Standard Laws of Physics

Arrangement of Hardware → Global B,E → Lorentz → Propagation for a time dt

Usual Beam Dynamics Code



Magnet Oriented

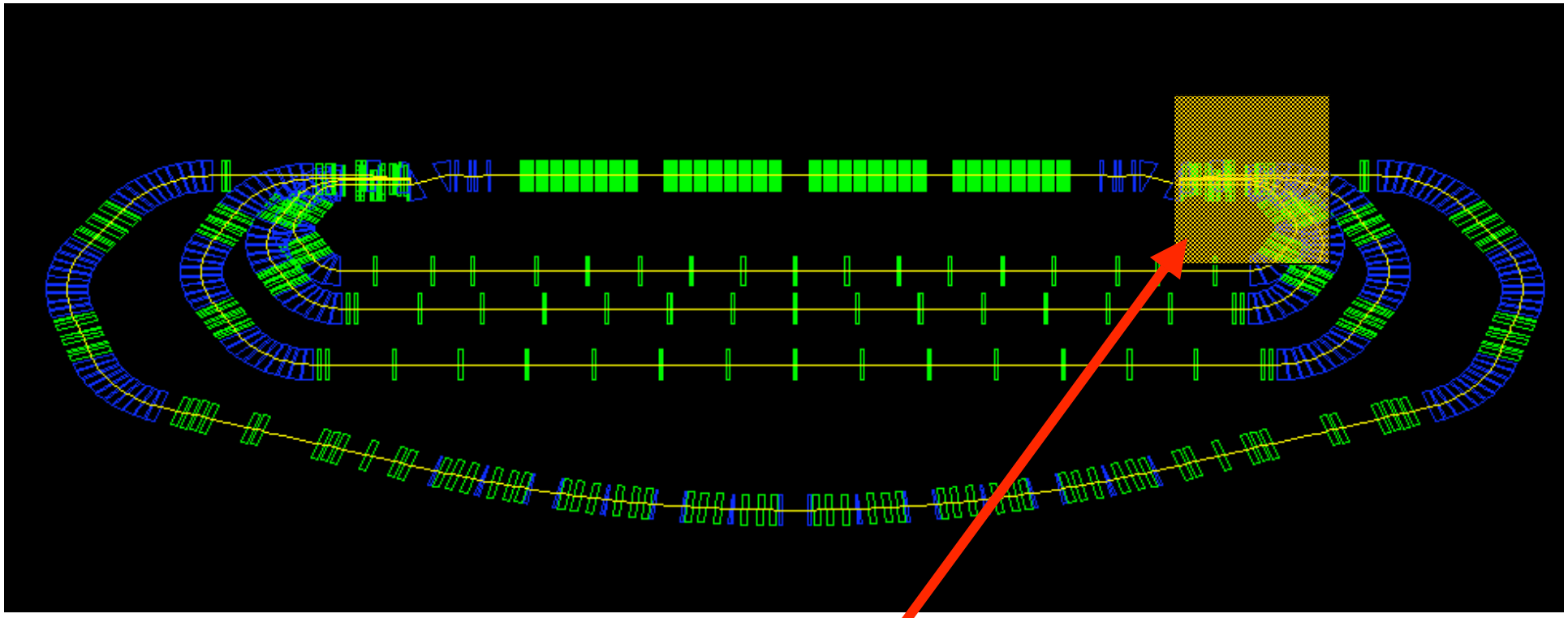
- Obviously map(s) attached to the magnets are central objects
- The B field is important but it is now attached to the magnet rather than a global object
- The two propagators shown in previous slide can exist in the same machine: dogbone, colliders, etc…
- What can we conclude?

Structures

1. Beam lines are actually sequences of the discretized “s” variables and not a sequence of magnet propagators as in standard codes.
2. I baptized the element of this sequence “fibres” inspired by Hirata since it is connected to the mathematical fibre bundles.
3. Fibres contains pointers to magnets and pointers to transformations to the local variables used at that particular “s” position: my so-called patches.

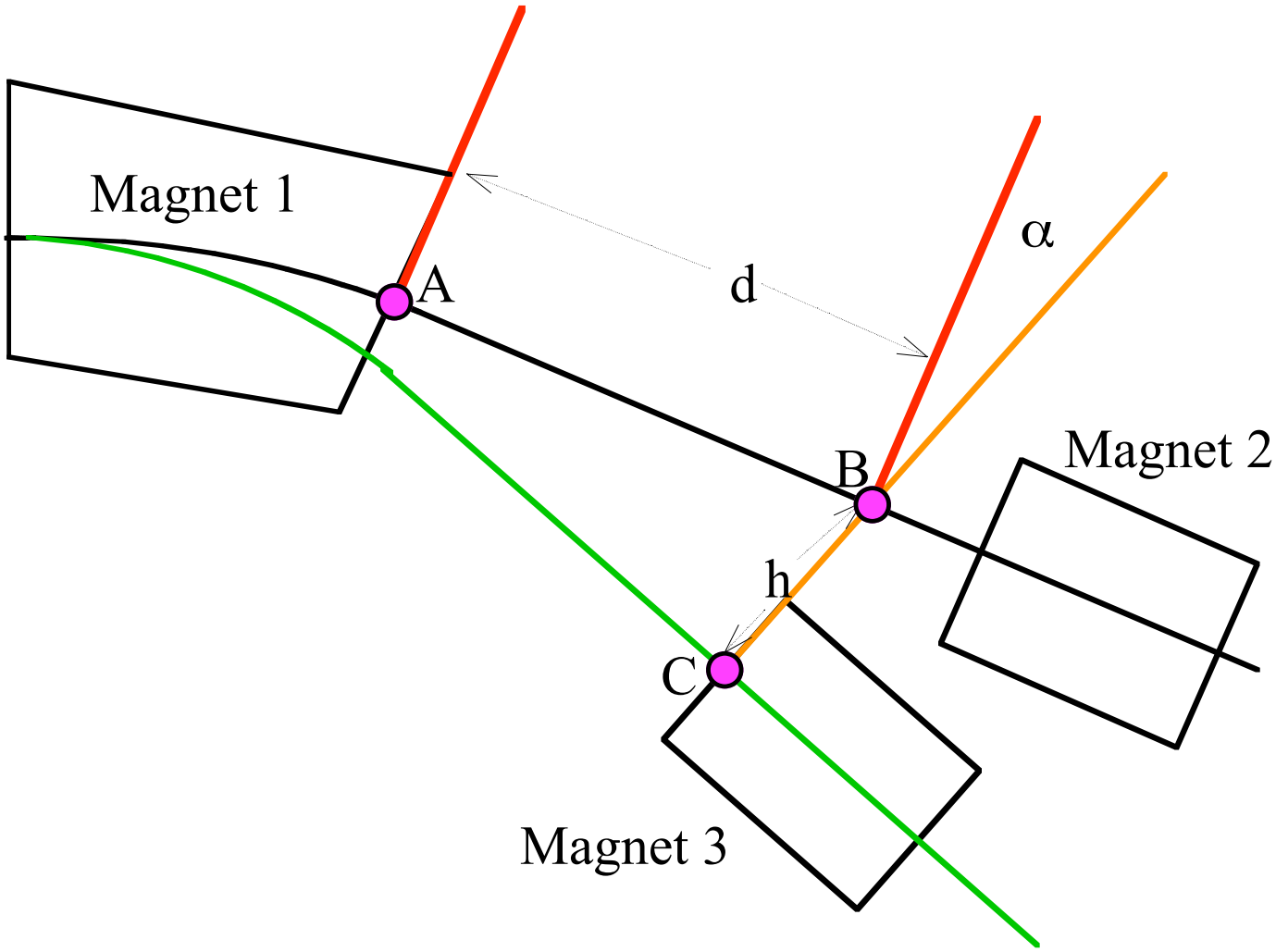
WHY?

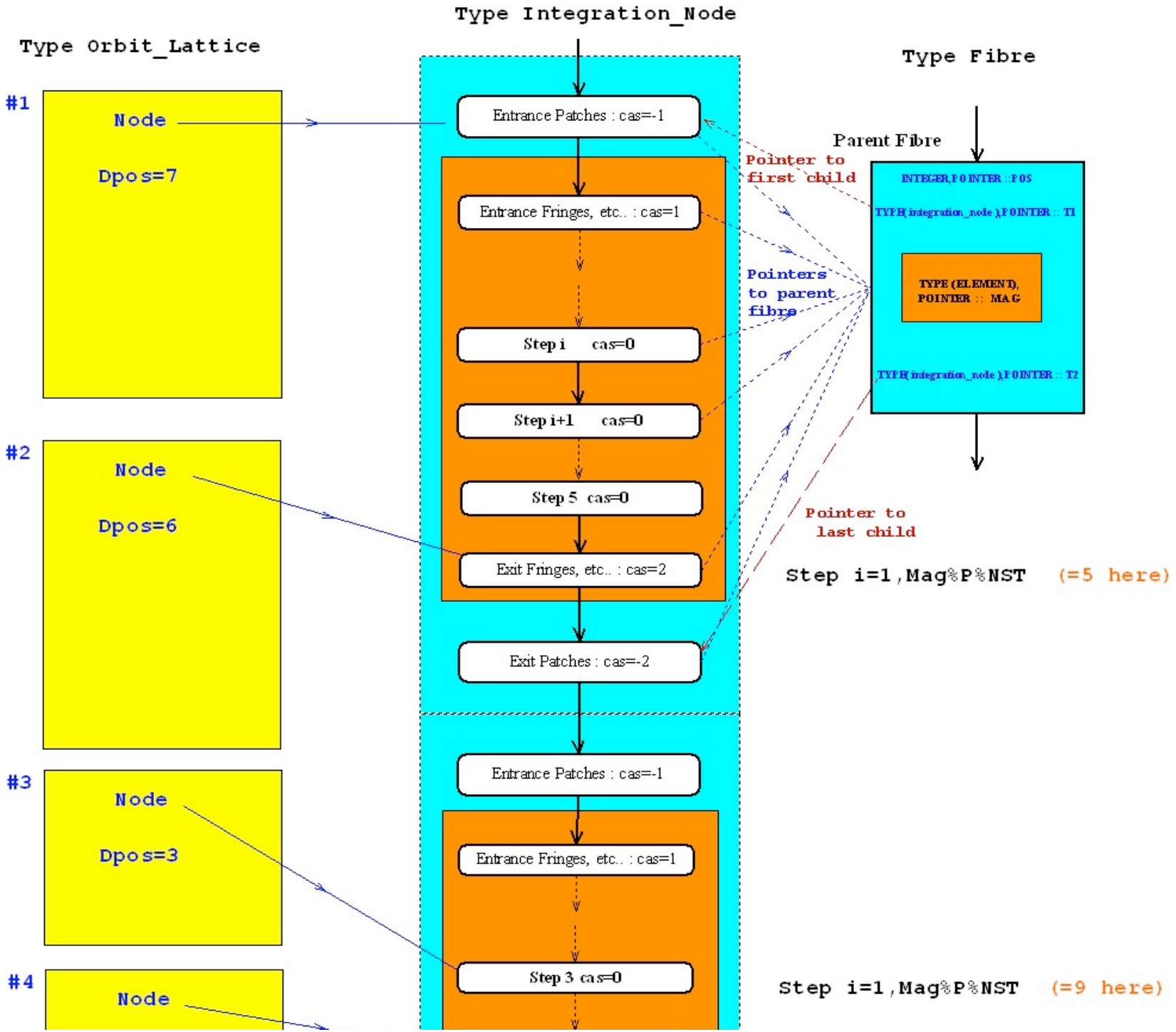
Example: Recirculator



Magnify this common region

Pictorial Justification




Type Integration_Node
Type Orbit_Lattice
Type Fibre
#1
Node
Dpos=7
#2
Node
Dpos=6
#3
Node
Dpos=3
#4
Node
Entrance Patches : cas=-1
Entrance Fringes, etc.. : cas=1
Step i cas=0
Step i+1 cas=0
Step 5 cas=0
Exit Fringes, etc.. : cas=2
Exit Patches : cas=-2
Entrance Patches : cas=-1
Entrance Fringes, etc.. : cas=1
Step 3 cas=0
Parent Fibre
Pointer to first child
INTEGER,POINTER :: POS
TYPE(integration_node),POINTER :: T1
TYPE (ELEMENT), POINTER :: MAG
TYPE(integration_node),POINTER :: T2
Pointers to parent fibre
Pointer to last child
Step i=1,Mag%P%NST (=5 here)
Step i=1,Mag%P%NST (=9 here)

Conclusions

- Most bizarre topologies permitted: recirculators, dogbones, colliders and multiple combinations of these horrors
- Fully 3d lattices like the quadrupole fibres of Mori and Machida. In PTC the survey command is essentially useless and definitely arbitrary. It is used for double checking geometric patching or entering standard lattices. (N.B. MAD and SAD have different convention for survey!)
- PTC is only one possible solution: Malitsky and Shishlo have argued that complex structure should be in flexible scripting languages (Python, etc…) and only the propagators (forward, reverse etc…) should be in compiled languages. OK: no fundamental objections and no personal skill to program that.
- MAD-X has PTC in it but it is not really capable of handling its structures properly. For example in MAD-X is a linked list of magnets (wrong) so that lattices functions remain properties of magnets which is truly false. Recirculators are not supported. Double ring of LHC is supported through hacking!
- So at present, besides Malitsky environment, I am not aware of compatible environments. SAD-X? Be my guest.