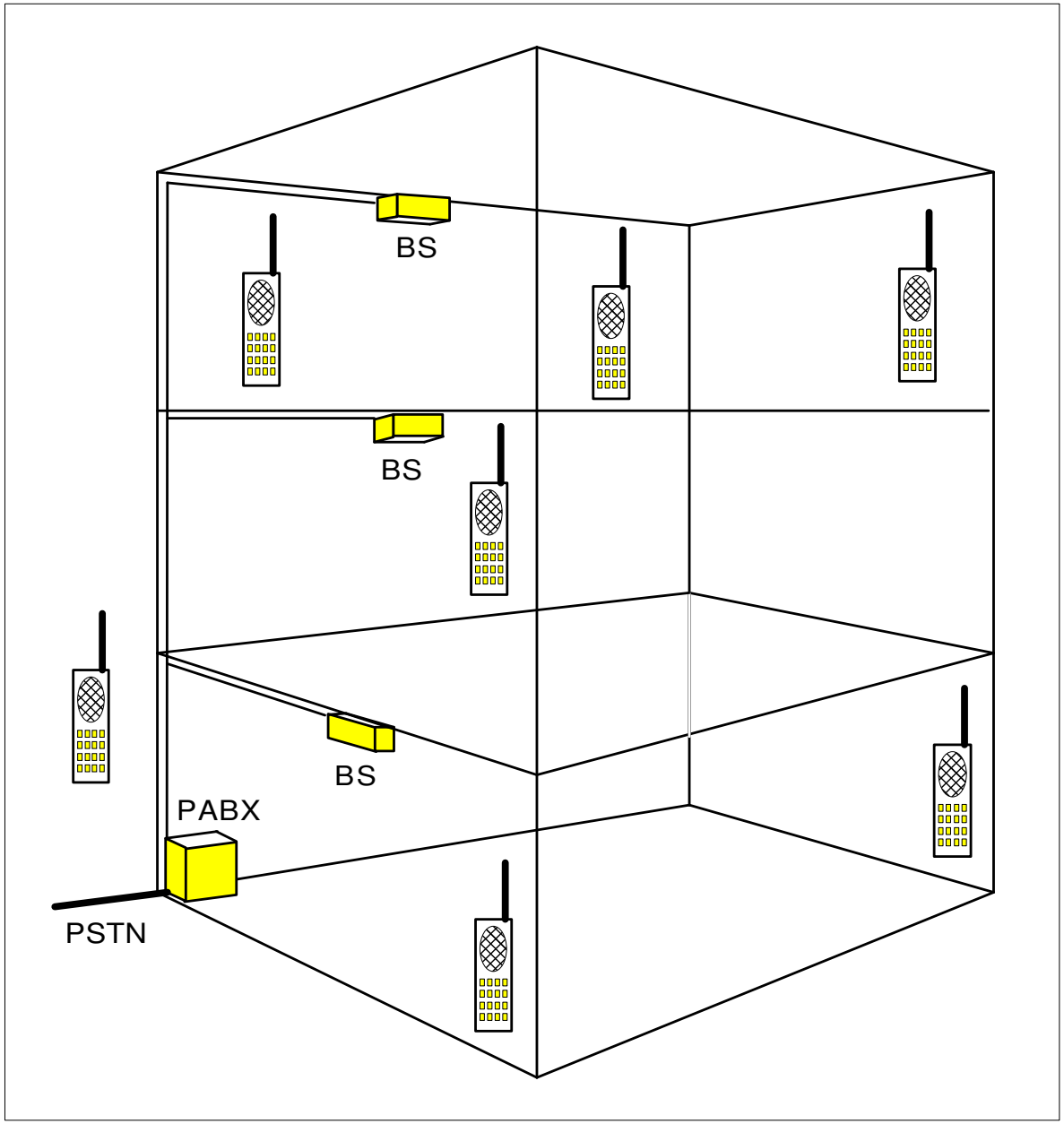
Optimal Planning of Digital Cordless Telecommunication Systems [a]

T. Frühwirth Ludwig-Maximilians-Universität München, Germany

P. Brisset École Nationale de l'Aviation Civile, France

[a]Work was done at ECRC, Munich, Germany



Using a Mobile Phone in a Building

The POPULAR prototype

Data :

- A blue-print of the building
- Information about the materials used for walls and ceilings

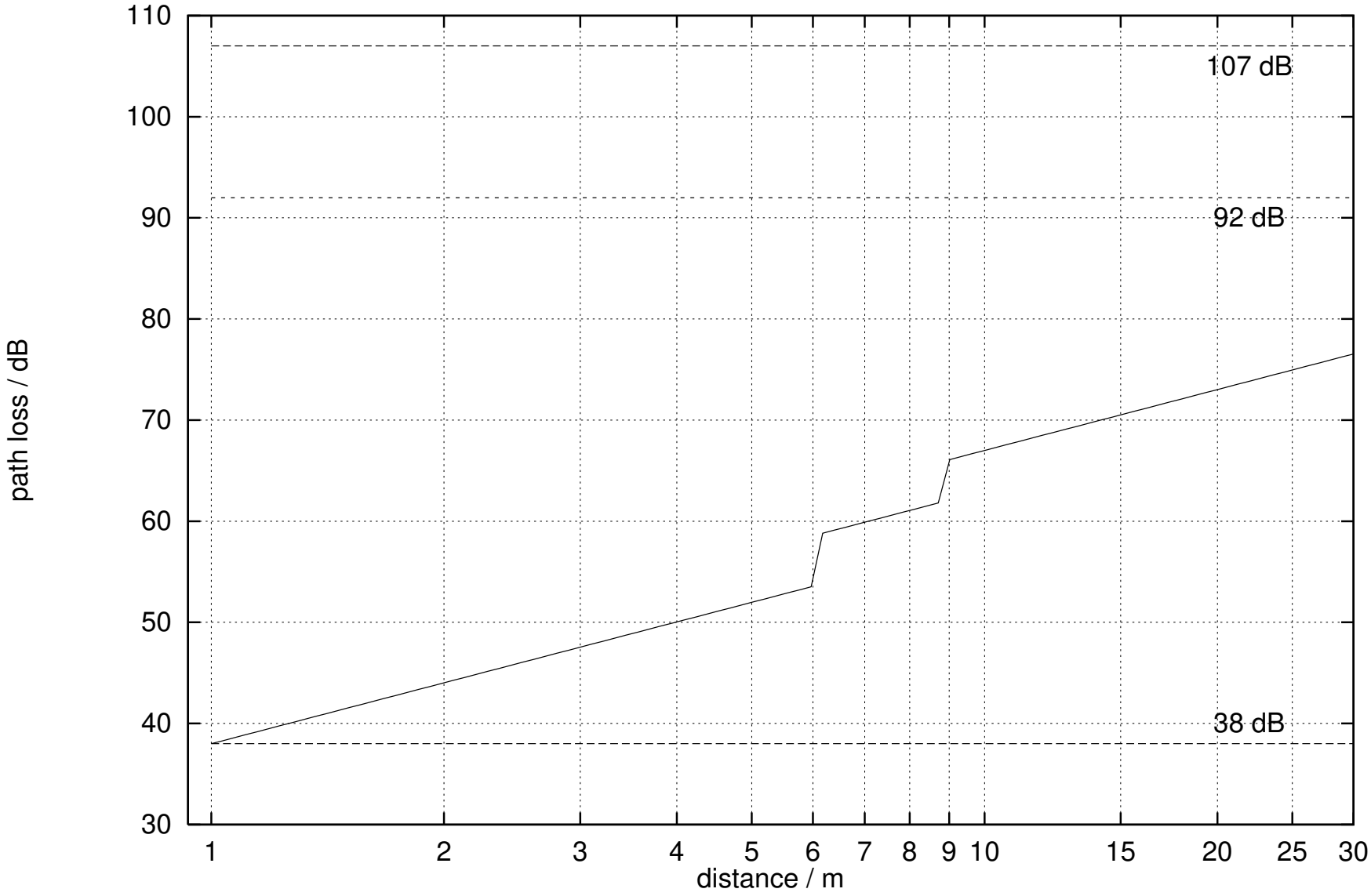
The problem :

- Placing senders to cover all the rooms in the building
- Computing the minimum number of senders needed

The solution :

- Using constraint technology

Propagation model: loss/distance



Propagation model (cont.)

$$L = L_{1m} + 10n \log_{10} d + \sum_i k_i F_i + \sum_j p_j W_j$$

L Total path loss in dB

L_{1m} path loss in $1m$ distance from the sender

n propagation factor

d distance between transmitter and receiver

k_i number of floors of kind i in the propagation path

F_i attenuation factor of one floor of kind i

p_j number of walls of kind j in the propagation path

W_j attenuation factor of one wall of kind j

Direct Encoding

A naive solution would be to

- Discretize the space in grid points P_i
- Express the relation (constraint) between senders S_j positions and signal level at each point P_i :
  $Signal(P_i) = \max_j(Signal(S_j) - Loss(S_j, P_i))$
- Express that the signal must be above a threshold at each point :
  $Signal(P_i) \geq Threshold$

It does not work because the relations are too complex to constrain senders positions.

Dual Problem

Since the propagation of a signal is not directional, sender and receiver can be exchanged.

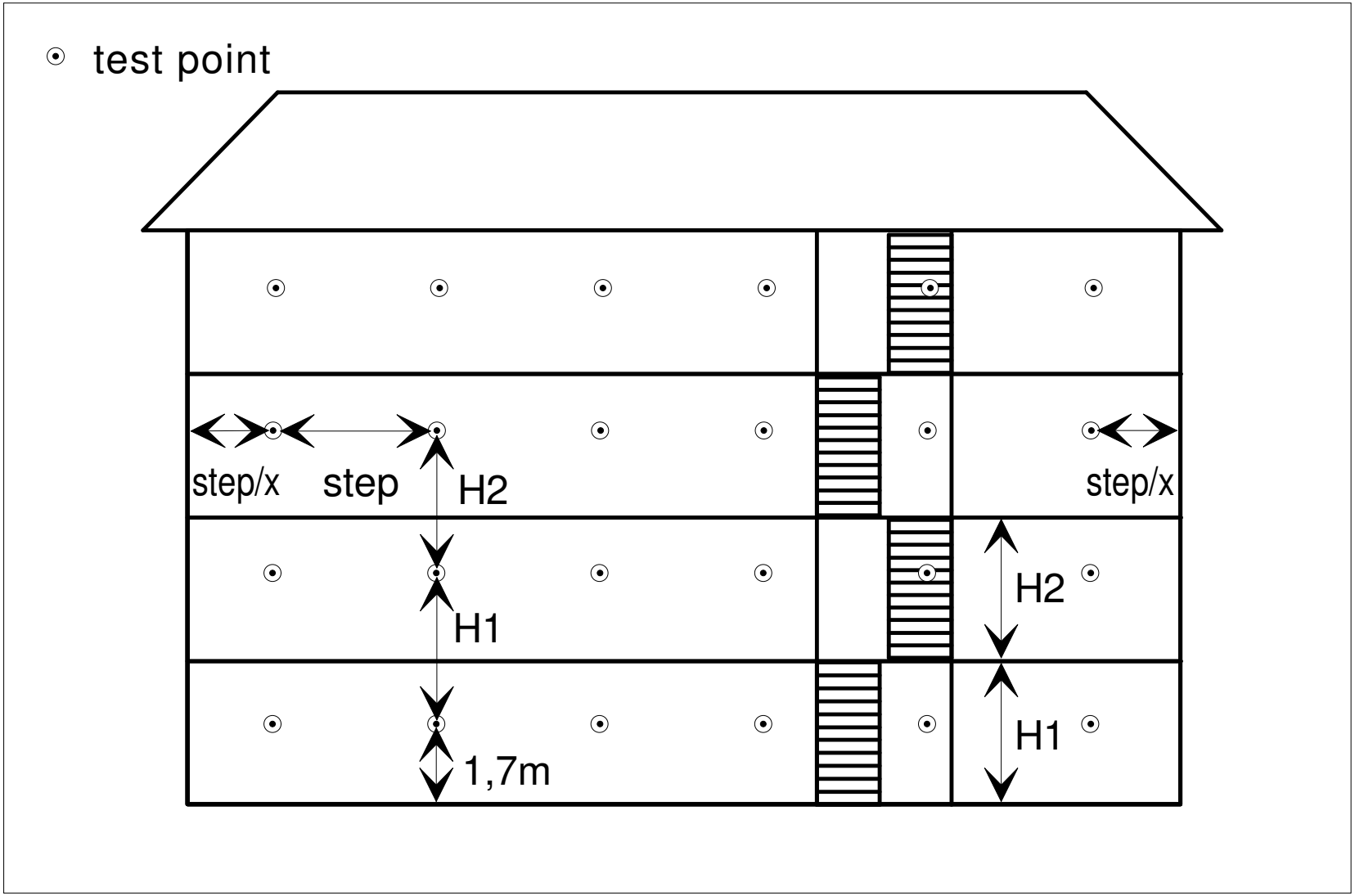
Therefore the two following properties are equivalent :

Each grid point is reached by the signal of one sender :
$\forall P_i \exists S_j \;\; P_i \in Covered(S_j)$

There is a sender in the neighbourhood of each grid point :
$\forall P_i \exists S_j \;\; S_j \in Covered(P_i)$

The dual problem is easier to solve because the $Covered(P_i)$ zones can be statically computed.

Grid of test points



Representation of Covered Surfaces

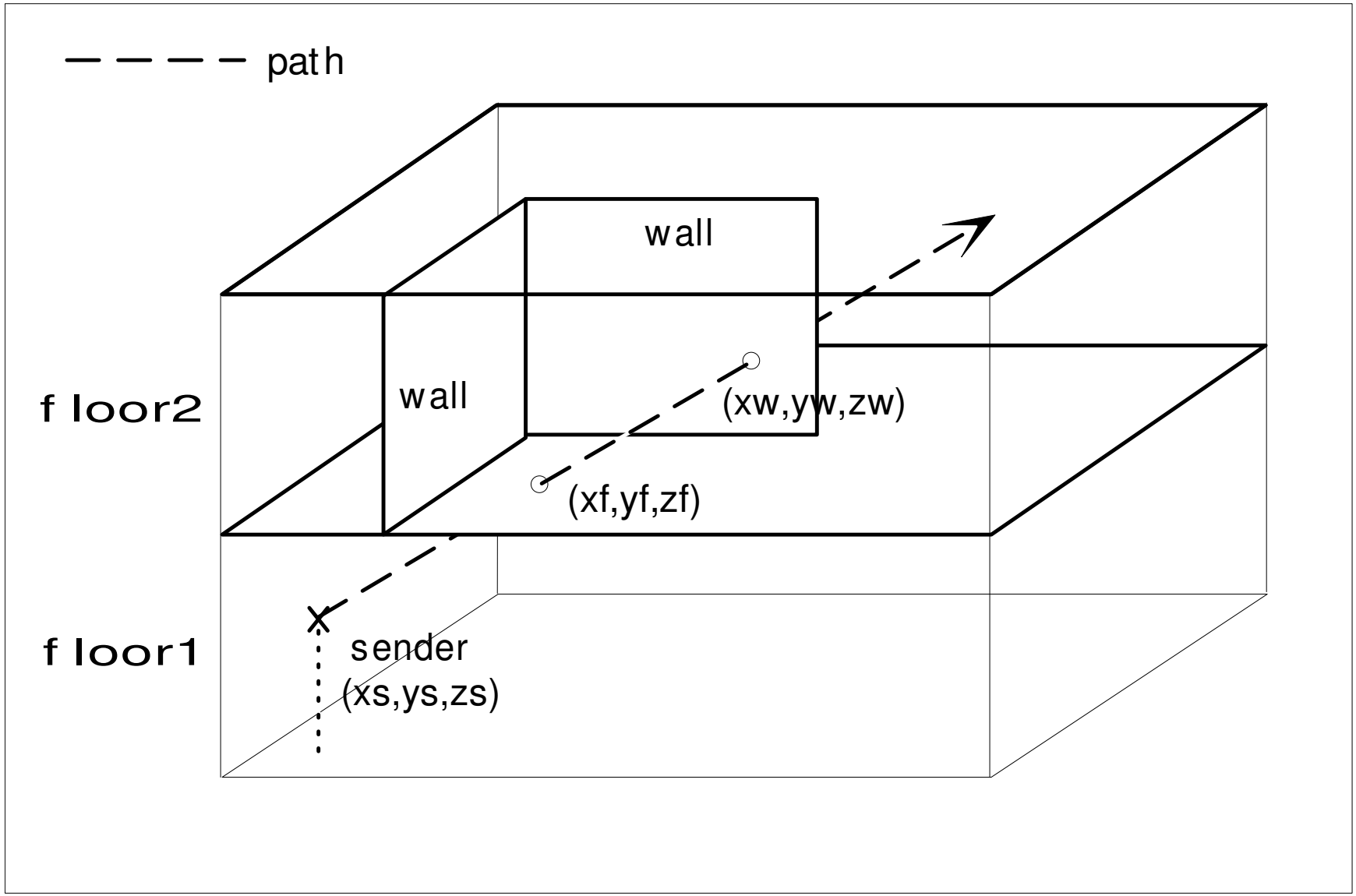
In order to express the constraint $S_j \in Covered(P_i)$, the $Covered(P_i)$ must be simple enough. It can be approximated by

- A rectangle
- A list of rectangles

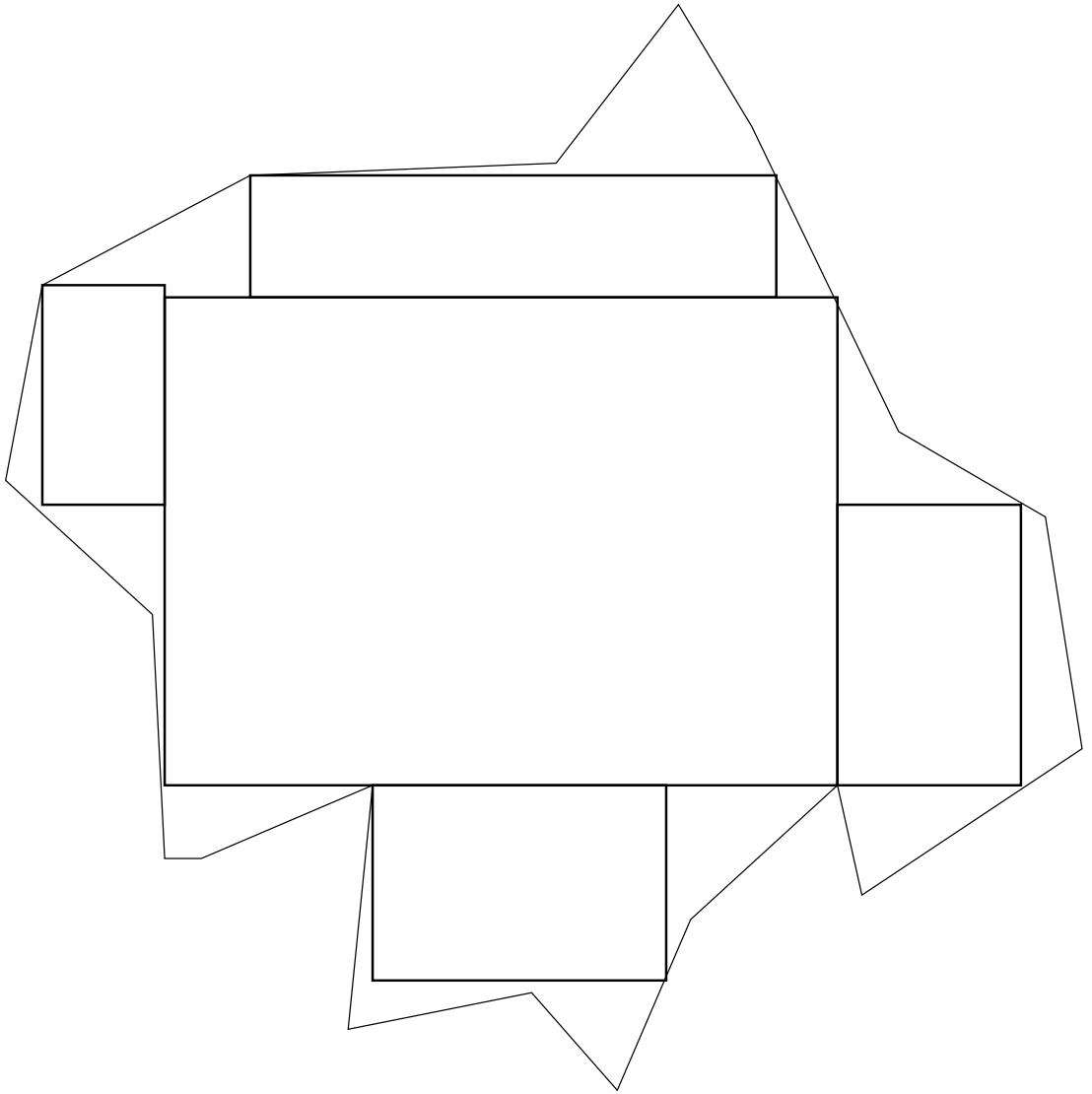
Algorithm

1. Compute the $Covered(P_i)$ zone by ray tracing for each P_i
2. Approximate $Covered(P_i)$
3. Set the constraints $S_j \in Covered(P_i)$
4. Do clever labeling

Ray Tracing



Approximation by a Union of Rectangles



Constraint Handling Rules

What? : A declarative language designed for writing user-defined constraints : a commited-choice language with multi-headed rules for rewriting the constraints into simple ones.

How ? : A library for the Prolog ECL^{i}PS^e system including

- a translator from constraint handling rules to Prolog code,
- a runtime for handling the constraint store.

CHR inside constraint

Rules for the `inside` constraint stating that a point is **inside** a rectangle

```
% inside((X0, Y0), (XLeftLow, YLeftLow)-(XRightUp, YRightUp))

inside(_, (Xm, Ym)-(XM, YM)) ==>
  Xm < XM, Ym < YM.

inside((X, Y), (Xm, Ym)-(XM, YM)) ==>
  Xm < X, X < XM, Ym < Y, Y < YM.

inside(XY, (Xm1,Ym1)-(XM1,YM1)), inside(XY, (Xm2,Ym2)-(XM2,YM2)) <=>
  Xm is max(Xm1,Xm2), Ym is max(Ym1,Ym2),
  XM is min(XM1,XM2), YM is min(YM1,YM2),
  inside(XY, (Xm,Ym)-(XM,YM)).
```

Extension to Union of Rectangles

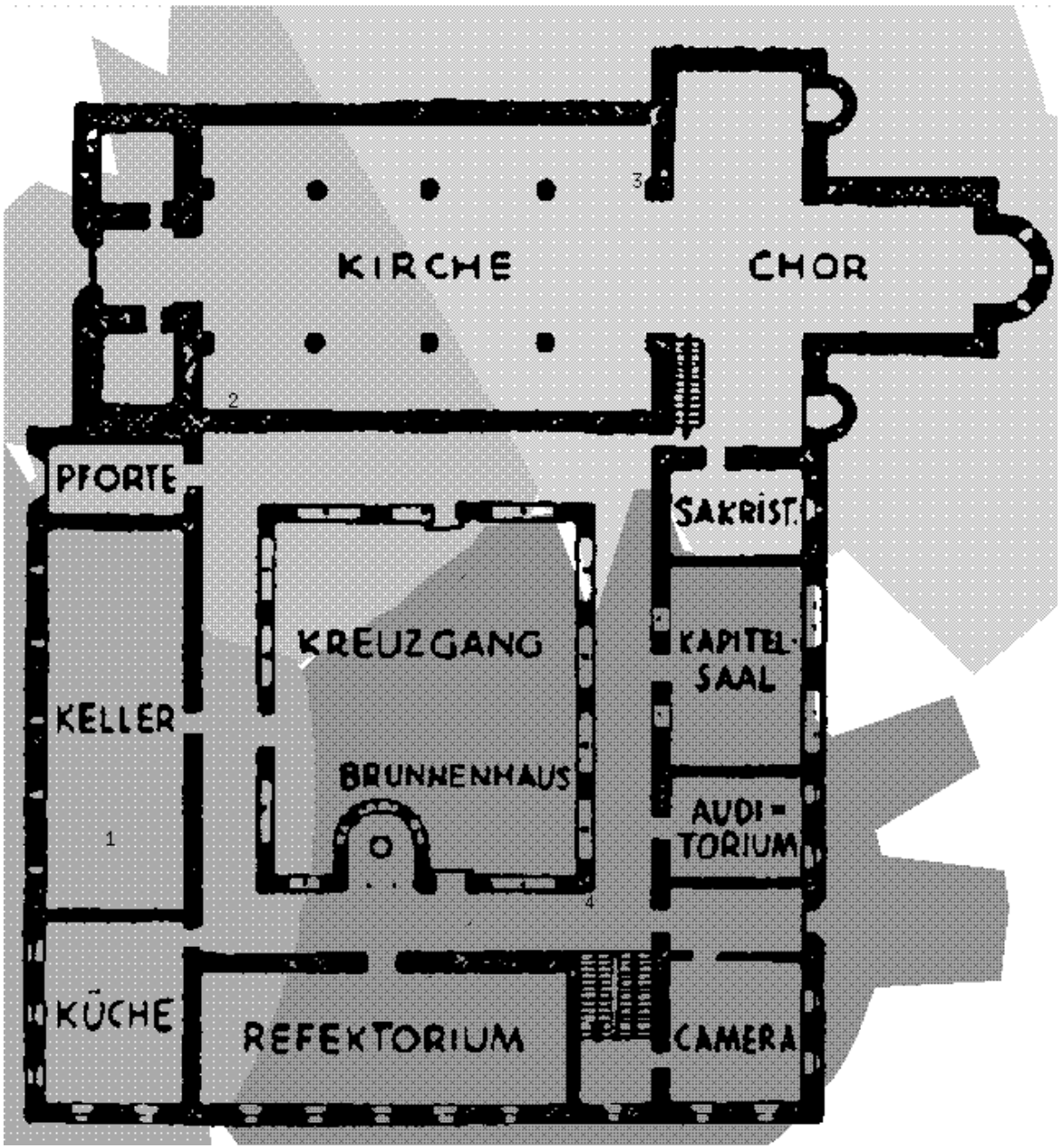
Rules for the `inside` constraint stating that a point is within a list of rectangles (a GEOMetrical object)

```
inside(S, L1), inside(S, L2) <=>
  intersect_geoms(L1, L2, L3),
  inside(S, L3).

intersect_geoms(L1, L2, L3) <=>
  setof(Rect, intersect_geom(L1, L2, Rect), L3).

intersect_geom(L1, L2, Rect) <=>
  member(Rect1, L1), member(Rect2, L2),
  intersect_rectangles(Rect1, Rect2, Rect).
```

Labeling

The constraint phase associates a sender to each $Covered(P_i)$ zone. The labeling phase has to choose the number and the positions of the senders. It is expressed by stating that as many senders as possible are equal.

```
equate_senders([]) <=> true.
equate_senders([S|L]) <=>
  ( member(S, L) or true ), % Try to equate a sender with others
  equate_senders(L).
```



A True Example

Conclusion

On this application, constraint technology (CHR) proves to

- have big expression power: the whole program for solving the problem is only a couple of hundred lines and required few man-months to be implemented.
- be flexible: the first prototype was easily extended from rectangles to union of rectangles, from 2-D to 3-D, ...
- be extensible: for example, restricting allowed senders locations to walls needs only one more inside constraint.
- be efficient: for a typical office building, an optimal placement is found within a few minutes (up to 25 base stations).