情報処理学会論文誌 Vol.59 No.4 1203–1214 (Apr. 2018)

フィーチャモデルの近似的解析による フィーチャ構成導出手法

岸 知二[1,a)] 野田 夏子[2]

受付日 2017年8月2日, 採録日 2018年1月15日

概要：フィーチャモデルからのフィーチャ構成導出は製品系列開発などで行われる重要な作業であるが，導出の計算量は NP 完全であり，大きなモデルからの導出や導出の繰返しはコストが高い．フィーチャモデルの構造を解析することによって，たとえば導出の繰返しを効率化したり，より小さなサイズのフィーチャ構成を導出したりすることができるが，構造の解析もまた高コストである．本稿では，構造の解析を低計算量で近似的に行い，それに基づいてフィーチャ構成を導出する手法について提案する．シミュレーションによる評価で，本手法がフィーチャ構成導出の繰返しや，小さなサイズのフィーチャ構成導出に有効性を持つことを確認した．

キーワード：フィーチャモデル，フィーチャ構成導出，ソフトウェアプロダクトライン開発

A Feature Configurations Derivation Method Based on Approximate Feature Model Analysis

Tomoji Kishi[1,a)] Natsuko Noda[2]

Received: August 2, 2017, Accepted: January 15, 2018

Abstract: Feature configurations derivation from feature model is an important activity in product-lines development. As the computational complexity of the derivation is NP-complete, derivations from large feature models or repetition of derivations are expensive. By analyzing feature model, we could repeat the derivation efficiently or derive smaller size of feature configuration. However, the analysis of feature model is also expensive. In this paper, we propose a feature configuration derivation method in which we analyze feature model approximately at lower cost. We evaluate the method in terms of simulation and the result shows the effectiveness of the method.

Keywords: feature model, feature configurations derivation, software product lines development

1. はじめに

フィーチャモデル（feature model，以下 FM）[13] は，製品群の共通性・可変性を外部から観測可能なフィーチャに照らして表現する可変性モデルであり，ソフトウェア製品系列（software product-lines，以下 SPL）開発 [6] などで使われる．FM 中では必須，任意，依存などといったフィーチャ間の制約が定義される．フィーチャ構成導出とは，FM とフィーチャ選択（FM 中のフィーチャのいくつかに対する選択あるいは非選択の指定）を与え，FM の制約とフィーチャ選択を満たすフィーチャ構成（集合）を求める作業であり，SPL 開発での製品導出 [9] などで行われる．フィーチャ構成導出の方法には SAT や CSP を用いた方法 [4], [18], [25] などがあるが，その計算量は NP 完全であり [20], [21]，大規模な FM からのフィーチャ構成導出や，導出の繰返しは高コストとなる．

フィーチャ選択に含まれないフィーチャ群に対する選択・非選択の割当て方法は複数ありうるため，一般に FM

[1] 早稲田大学
Waseda University, Shinjuku, Tokyo 169–8555, Japan
[2] 芝浦工業大学
Shibaura Institute of Technology, Minato, Tokyo 108–8548, Japan
[a)] kishi@waseda.jp

の制約とフィーチャ選択を満たすフィーチャ構成は複数あり，フィーチャ構成導出ではそれらの中の 1 つが導出される．製品導出ではこうして導出されたフィーチャ構成に基づき，たとえばソースコード中でコンパイラ制御文を用いて表現された選択肢を選ぶなど，成果物中の可変性の解決に利用する．しかしながら FM で表現される可変性構造とそうした成果物中の可変性構造の間には不整合が存在することが報告されている [16], [19]．著者らもこうした理由から，得られたフィーチャ構成に基づく可変性の解決が適切に行われずフィーチャ構成導出を繰り返すことがあった．あるいは，フィーチャ数が多いと実装やテストの量が増えるため，選択したフィーチャ以外の余分なフィーチャをできるだけ含まない小さなサイズのフィーチャ構成を導出したいこともある．

FM の構造を解析することで，その選択・非選択がフィーチャ選択に影響されないフィーチャ群や，それを非選択としても他のフィーチャの選択・非選択に影響を及ぼさないフィーチャ群を識別することができる．こうした解析に基づいてフィーチャ構成導出を行うことで，導出の繰返しや小さなサイズのフィーチャ構成の導出をより効果的に行うことができる．こうした解析を厳密に行うには，たとえば FM のスライシング [1] などを利用することが考えられるが，その計算量は NP 困難である [15]．また具体的な利用方法は知られていない．

本稿では，FM の構造の解析を近似的に行い，それに基づいてフィーチャ構成を導出する手法について提案する．容易に特定できるフィーチャ群のみを識別するため，識別のための計算量は低く（フィーチャ数の 3 乗のオーダ）現実的である．一方厳密な識別をしないため，たとえばフィーチャ選択に影響されないフィーチャであっても識別できない可能性があるが，シミュレーションによる評価で，フィーチャ構成導出の繰返しや小さなサイズのフィーチャ構成導出に有効であることを確認した．なお本稿は著者らの過去の報告 [14] の内容を再整理するとともに，提案手法を厳密に定義し，さらに評価を加えたものである．

本稿は以下のように構成される．2 章ではフィーチャ構成導出と FM の構造について説明し，3 章では FM の構造解析に基づくフィーチャ構成導出の有用性と課題について述べる．4 章で近似的解析に基づくフィーチャ構成導出手法を提案し，5 章で本手法の有用性の評価を行う．6 章で議論を行うとともに，7 章では関連研究に触れ，8 章で締めくくる．

2. フィーチャ構成導出

本章では FM とフィーチャ構成導出について説明した後，フィーチャ構成導出に関わる FM の構造について指摘する．

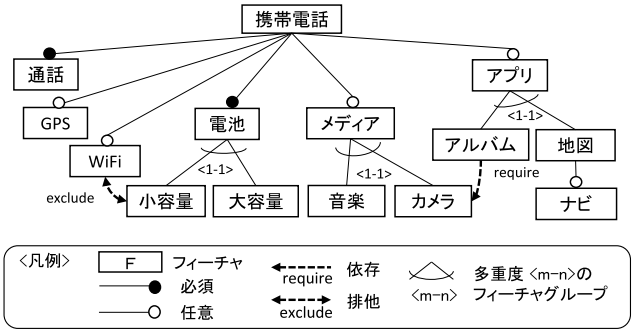

図 1 FM の例
Fig. 1 An example of FM.

2.1 フィーチャモデル（FM）

FM は外部から観測可能なフィーチャに照らして製品群の可変性を表現する可変性モデルである [13]．**図 1** に FM の記述例を示す．FM 中では必須（親が選ばれれば必ず選ばれる），任意（親が選ばれても選ぶことは任意），依存（依存先が選ばれないと選べない），排他（両者を同時に選べない），フィーチャグループ [7]（親が選ばれれば多重度の範囲でグループ中のフィーチャが選ばれる）などの制約が定義される．なお依存と排他をクロスツリー制約（cross tree constraint，以下 CTC）と呼ぶ．

2.2 フィーチャ構成導出

フィーチャ構成が FM の制約を満たすとき，それを正しいフィーチャ構成と呼ぶ．フィーチャ構成導出は，FM とフィーチャ選択（FM 中のフィーチャのいくつかに対する選択あるいは非選択の指定）が与えられたとき，フィーチャ選択を満たす正しいフィーチャ構成を求める作業である．フィーチャ選択に含まれないフィーチャ群に対する選択・非選択の割当て方法は複数ありうるので，上記を満たすフィーチャ構成は一般に複数あり，それらの中から 1 つを導出する．上記を満たすフィーチャ構成が存在しない場合には，ないことを出力する．

フィーチャ構成導出の方法としては，SAT や CSP を使った方法 [4], [18], [25] などが提案されており，本稿でも類似の方法を用いる．**表 1** は FM の制約を制約ソルバ Sugar [24] での制約記述（群）に対応づけた例である．変数はフィーチャに対応し，値 1，0 はそれぞれ選択・非選択を表す．“(eq *t1 t2*)” は等価，“(imp *f1 f2*)” は含意，“(weighted sum ((*w1 v1*) .. (*wn vn*)), [ge|le], *t*)” は，*w1*v1*+ · · · +*wn*vn* が ge なら t 以上，le なら t 以下という制約を表す．フィーチャの選択・非選択は “(eq f 1)” あるいは “(eq f 0)” で表す．制約を満たす構成があれば，その構成（変数への値の割当て）の 1 つが示される．

2.3 フィーチャモデルの構造

図 1 の FM において {“カメラ”，“地図”} というフィー

表 1 FM と CSP の制約記述の対応
Table 1 Mapping between FM and CSP description.

FM の要素	CSP の制約記述(Sugar[24]の場合)
r はルート	(eq *r* 1)
m は *p* の必須の子フィーチャ	(imp (eq *p* 0) (eq *m* 0)) (imp (eq *p* 1) (eq *m* 1))
o は *p* の任意の子フィーチャ	(imp (eq *p* 0) (eq *o* 0)) (imp (eq *p* 1) (or (eq *o* 1) 1))
p は *g1..gn* を持つ多重度 <*m-n*> のフィーチャグループ	(imp (eq *p* 0) (and ((eq *g1* 0) … (eq *gn* 0)))) (imp (eq *p* 1) (or ((eq *g1* 1) … (eq *gn* 1)))) (imp (eq *p* 1) (weightedsum ((1 *g1*) .. (1 *gn*)) ge *m*)) (imp (eq *p* 1) (weightedsum ((1 *g1*) .. (1 *gn*)) le *n*))
f1 require *f2*	(imp (eq *f1* 1) (eq *f2* 1))
f1 exclude *f2*	(not (and (eq *f1* 1) (eq *f2* 1)))

表 2 導出されうるフィーチャ構成
Table 2 Derivable feature configurations.

	携帯電話	通話	GPS	WiFi	電池	小容量	大容量	メディア	音楽	カメラ*	アプリ	アルバム	地図*	ナビ	構成サイズ
#0	1	1	0	0	1	0	1	1	0	1	1	0	1	1	9
#1	1	1	0	0	1	1	0	1	0	1	1	0	1	0	8
#2	1	1	0	0	1	1	0	1	0	1	1	0	1	1	9
#3	1	1	0	0	1	0	1	1	0	1	1	0	1	0	8
#4	1	1	0	1	1	0	1	1	0	1	1	0	1	0	9
#5	1	1	0	1	1	0	1	1	0	1	1	0	1	1	10
#6	1	1	1	0	1	1	0	1	0	1	1	0	1	1	10
#7	1	1	1	0	1	0	1	1	0	1	1	0	1	0	9
#8	1	1	1	0	1	0	1	1	0	1	1	0	1	1	10
#9	1	1	1	0	1	1	0	1	0	1	1	0	1	0	9
#10	1	1	1	1	1	0	1	1	0	1	1	0	1	1	11
#11	1	1	1	1	1	0	1	1	0	1	1	0	1	0	10

チャ選択が与えられた状況を考える．これは“カメラ”と“地図”を選択することを指定し，非選択の指定はなく，それ以外のフィーチャは選択・非選択どちらでもよいという意味となる．このとき導出されうるフィーチャ構成は 12 個ある（**表 2**）．

表の各行は構成に対応し，最左列に識別番号を示す．列はフィーチャに対応する．コラムの 1 と 0 はその構成で該当するフィーチャが選択あるいは非選択であることを示す．最右列は，後述するフィーチャ構成のサイズ（3.2）である．なお選択されたフィーチャに∗印をつけている．

ここで，FM の構造に関して以下の点に注目する．

(1) 3 つの部分への分割

“通話”と“電池”はルートフィーチャ（“携帯電話”）と必須の関係で結ばれ必ず選択される．また，フィーチャ選択中のフィーチャ（“地図”と“カメラ”）を含む，“メディア”と“アプリ”以下のサブ木と，含まない“GPS”，“WIFI”，“電池”以下のサブ木は相互に独立していて影響を及ぼさない．これは FM の性質上，あるフィーチャの選択・非選択は，①親フィーチャの選択・非選択，②フィーチャグループメンバの場合は同じフィーチャグループ中の他のフィーチャの選択・非選択，③CTC でつながれたフィーチャの選択・非選択によってのみ制約されるからである．表 2 の構成は，**図 2** のように，必ず選択される部分，フィーチャ選択に影響される部分，フィーチャ選択に影響されない部分の各構成の直積と理解することができる．なお，図中の共通部分などの名称については後述する．

図 2 フィーチャ構成の構造
Fig. 2 Feature configurations' structure.

(2) 非選択が他に影響を及ぼさないフィーチャの存在

“ナビ”を選択するためには，親フィーチャの選択が前提となるが，このフィーチャの非選択は他のフィーチャの選択・非選択に影響を及ぼさない．“GPS”も同様である．もしもこれらのフィーチャが子孫フィーチャを持っていても，それらがフィーチャ選択中に含まれず，他のフィーチャと CTC による関連を持たなければ，その部分全体を非選択としても他に影響を及ぼさない．

3. 構造に基づくフィーチャ構成導出と課題

本章では，前章で指摘した FM の構造をフィーチャ構成導出に有効に利用できることを指摘するとともに，その際の課題について述べる．

3.1 部分ごとのフィーチャ構成導出

FM を 3 つの部分に分割することにより，フィーチャ選択の影響を受ける部分からのフィーチャ構成導出と，影響を受けない部分からのフィーチャ構成導出を独立して行うことができる．こうしたフィーチャ構成導出が有用なことがある．

まずフィーチャ構成を 3 つの部分の構成に分割してその直積としてとらえることで，各構成は相対的に小さくなりフィーチャ構成の理解を容易にすることができる．またフィーチャ構成導出を繰り返す場合には，自分の関心のあるフィーチャを含む部分からの導出のみを繰り返せばよく，効率的である．あるいはフィーチャ構成のリストアップ [10] を行う際にも，それぞれの部分からのリストアップを行えばよいので，導出回数が減る．たとえば上述の例では 12 通りの構成が存在するが，分割すればそれぞれ 2 回

と 6 回の計 8 回の導出でリストアップできる．

FM を分割してそれぞれからフィーチャ構成の導出を行っても，全体のフィーチャ構成を読み解くためにはその組合せを理解する必要がある．したがって分割して一見フィーチャ構成の数が少なくなるように見えても，本質的なフィーチャ構成数が変わるわけではない．しかしながら，上記のような利用においては部分ごとのリストアップで十分なこともあり，分割することでリストアップできるケースが増えることは実用上意味がある．

3.2 小さなサイズのフィーチャ構成の導出

フィーチャ構成中のフィーチャの数をフィーチャのサイズとする．非選択が他に影響を及ぼさないフィーチャ群を識別し，それらのフィーチャ群を非選択にしてしまうことで，小さなサイズのフィーチャ構成を導出できる．こうした小さなサイズのフィーチャ構成が有効な場合がある．

導出されたフィーチャ構成に基づいて，実装やテストを行う際には，大きなサイズのフィーチャ構成は，実現やテストをするフィーチャが増えることになり開発量やテスト量が多くなることが想定される．サイズの違いは選択を指定していないフィーチャ群の過多に依存しており，そういうフィーチャ群は余分と考える場合である．

また，ソフトウェアサプライチェーンなどで，フィーチャ選択を徐々に進めていく段階的構成 [8] が提案されている．ここではフィーチャ選択を 1 度に決定せず複数のステップで多段的にフィーチャ選択を行う．これはフィーチャ構成導出において暫定的に “不明” のラベルを付けることに対応する [4]．この際，上述した余分な部分を “不明” として小さな決定をすることで，後続ステップでの選択肢を広げることができる．たとえばあるステップで “WiFi” の選択・非選択を決める必要がないなら，無理にそれを決定せずに後続のステップで選択・非選択を選べるようにして自由度を高めることが望ましいと考える場合である．

3.3 FM の構造解析の課題

このように FM の構造を解析し，3 つの部分に分割したり非選択が他に影響を及ぼさないフィーチャ群を識別したりすると，フィーチャ構成導出に有用に活用できる場合があるが，その識別を厳密に行う場合の計算量は大きい．たとえば FM のスライシング [1] を利用することなども考えられるが，その計算量は NP 困難である [15]．またその具体的な方法が知られているわけではない．したがって，厳密な識別でなくても，低計算量で有効な FM の構造解析の手法の明確化が望まれる．

4. 提案手法

本章では，前章で指摘した課題を改善するための手法について提案する．

4.1 全体像

提案手法の目的は，低計算量で FM の構造解析を行い，それに基づくフィーチャ構成導出を行うことである．構造解析では，2.3 節で述べた 3 つの部分への分割と，非選択が他に影響を及ぼさないフィーチャ群の識別を近似的に行う．フィーチャ構成の導出においては，3 つの部分への分割に基づいた部分ごとのフィーチャ構成導出の手法と，非選択が他に影響を及ぼさないフィーチャ群の識別に基づいた小さなフィーチャ構成の導出の手法を示す．この両者を組み合わせることで，部分ごとに小さなフィーチャ構成を導出することもできる．

4.2 3 つの部分への分割

4.2.1 共通部分・コア部分・直交部分

FM とフィーチャ選択が与えられたときに，FM を以下の 3 つの部分に分割して理解することができる．

- 共通部分：フィーチャ選択にかかわらず，その FM から導出可能なすべての正しいフィーチャ構成に必ず含まれるフィーチャ群から構成される部分．
- コア部分：与えられたフィーチャ選択中のフィーチャの選択・非選択によってその選択・非選択が影響を受けるフィーチャ群から構成される部分．
- 直交部分：与えられたフィーチャ選択中のフィーチャの選択・非選択によってその選択・非選択が影響を受けないフィーチャ群から構成される部分．ただし共通部分中のフィーチャ群は除く．

本手法ではこれらを近似的に識別する．まず共通部分に関してはルートフィーチャから必須でたどれるサブ木部分だけを識別する（共通クラスタと呼ぶ）．次に FM を共通クラスタと直接親子関係を持つフィーチャを頂点とするサブ木に分割する．コア部分と直交部分はこれらのサブ木単位で識別する（それぞれコアクラスタ，直交クラスタと呼ぶ）．以下に手順を説明する．

4.2.2 共通クラスタの識別

処理の都合上，フィーチャグループの頂点を表すダミーのフィーチャを挿入する．これは FM の表現形式である SXFM に採用されている方法 [17], [23] と同様である．

図 3 では図 1 の FM に対して，ダミーのフィーチャ “G0”，“G1”，“G2” が挿入されている．なお最終的に導出するフィーチャ構成にはこれらのダミーのフィーチャは含めない．

図 4 は共通クラスタの識別アルゴリズムを疑似言語で記述したものである．ここではルートフィーチャから必須でつながるフィーチャ群を見つけ共通クラスタとする．依存などを考慮するとこれ以外のフィーチャが共通部分に含まれる場合もあるが，共通クラスタには含めない．本アルゴリズムでは同時に共通クラスタと親子関係でつながるフィーチャ（クラスタの頂点フィーチャ）も求める．

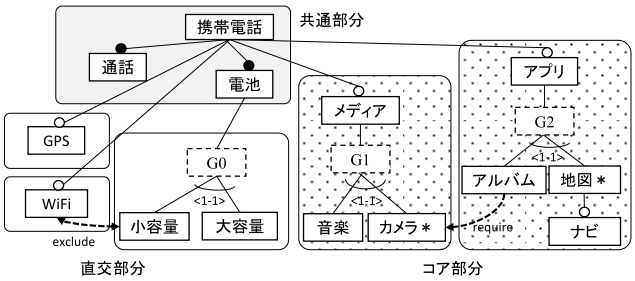

図 3 FM のクラスタ分割の例
Fig. 3 An example of FM clustering.

```
input: フィーチャモデル FM
output: 共通クラスタに含まれるフィーチャの集合: FCOM
        クラスタの頂点フィーチャの集合: FTOP
algorithm:
let FCOM:= {root}; FTOP := {} ; FCHK := {root} ;
while (FCHK != empty) do {
    FCHK 中からフィーチャ f を取り出す;
    forall 子フィーチャ cf of f do {
        if (cf が必須フィーチャ) {
            FCOM := {cf} ∪ FCOM; FCHK := {cf} ∪ FCHK ;
        }
        else {
            FTOP :={ cf } ∪ FTOP;
        }
    }
}
```

図 4 共通クラスタの識別
Fig. 4 Identification of common cluster.

ルートフィーチャから親子関係をたどるため，計算量はフィーチャ数を N とすると $O(N)$ である．

図 3 では，“携帯電話”，“通話”，“電池” が共通クラスタに含まれ，“GPS”，“WiFi”，“G0”，“メディア”，“アプリ” がクラスタの頂点となる．

4.2.3 コアクラスタと直交クラスタの識別

図 5 はコアクラスタと直交クラスタの識別アルゴリズムである．ここではフィーチャ選択中のフィーチャを含むクラスタ，ならびにそのクラスタと CTC で接続されて相互に影響しうるクラスタをコアクラスタとし，それ以外を直交クラスタと判断する．なお CTC のうち依存は有向，排他は無向（あるいは双方向）であるが，両者とも無向として扱っている．これは CTC の向きと影響の有無は必ずしも一致しない（方向にかかわらず影響を持つことも持たないこともありうる）ため，安全側に倒して（方向にかかわらず影響を持ちうるという立場で）識別するためである．

アルゴリズム中ではクラスタの頂点フィーチャの集合 V と，CTC によるクラスタの接続関係の集合 E によって構成されるグラフ $G = (V, E)$ を作成し，フィーチャ選択中に含

```
input: フィーチャモデル FM, フィーチャ選択 FC
       共通クラスタに含まれるフィーチャの集合: FCOM
       クラスタの頂点フィーチャの集合: FTOP
output: コアクラスタの頂点フィーチャの集合: FTOPcore
        直交クラスタの頂点フィーチャの集合: FTOPortho
algorithm:
let FTOPcore:={}, FTOPortho:={}; FTOPchoice:={};CCTC:= {};
forall フィーチャ f in FC do
    f から親を辿って到達する FTOP 中の tf を見つける;
    FTOPchoice := {tf} ∪ FTOPchoice;
}
forall CTC ctc in FM do {
    f0 と f1 を ctc の両端のフィーチャとする;
    if (f0 もしくは f1 が FCOM に含まれる) continue ;
    f0 から親を辿って到達する FTOP 中の tf0 を見つける;
    f1 から親を辿って到達する FTOP 中の tf1 を見つける;
    CCTC := {(tf0, tf1)} ∪ CCTC ;
}
G = (V, E) を V=FTOP, E=CCTC であるグラフとする;
FTOPcore := FTOPchoice ; FCHK := FTOPchoice ;
while FCHK が空でない do {
    フィーチャ f を取り出す ;
    forall 接続関係 r in CCTC do {
        if (r の一方の端のフィーチャが f かつ
            もう一方の端が FTOPcore に含まれない) {
            FTOPcore :={もう一方の端のフィーチャ} ∪
                        FTOPcore ;
            FCHK := {もう一方の端のフィーチャ} ∪ FCHK ;
        }
    }
}
FTOPortho := FTOP \ FTOPcore ;
```

図 5 コアクラスタと直交クラスタの識別
Fig. 5 Identification of core and orthogonal clusters.

まれるフィーチャを含むクラスタの頂点集合 $FTOPchoice$ を起点として，E 中のエッジ（接続関係）でたどれる V 中のノードの集合を求め，それをコアクラスタの頂点集合とする．

クラスタ単位で判断するためコアクラスタはコア部分より大きめに識別され，直交クラスタはそれ以外なので直交部分より小さめに識別される．

フィーチャ選択の親をたどる処理は $O(N^2)$ であり，CTC の数は $O(N^2)$ なのでその両端の親をたどる計算量は $O(N^3)$ である．グラフ G の V（クラスタの数）は $O(N)$，E（クラスタを結ぶ CTC の数）は $O(N^2)$ なので，コアクラスタの頂点集合を求める計算量も $O(N^3)$ である．

図 3 ではコアクラスタとして “メディア”，“G1”，“音

```
input: フィーチャモデル FM, フィーチャ選択 FC
       共通クラスタに含まれるフィーチャの集合: FCOM
output: 付加サブ木の頂点フィーチャの集合: FOSUB
algorithm:
let FOSUB := {};
let FCHK := {} ; //親をたどるフィーチャを格納
FCHK := FC ;
forall CTC ctc in FM do {
  foreach ctc に参加するフィーチャ f do {
    FCHK := {f} ∪ FCHK ;
  }
}
forall フィーチャ f in FCHK do {
  f にマークする;
  while (f が親フィーチャ p を持つ) do {
    if (p が FCOM に含まれる or p にマークがある) break;
    f にマークする;
    f := p ;
  }
}
forall 任意フィーチャ of in FM do {
  if (of がマークされていない) FOSUB := {of} ∪ FOSUB
}
```

図 6 付加サブ木の識別
Fig. 6 Identification of optional sub trees.

楽”，“カメラ”，“アプリ”，“G2”，“アルバム”，“地図”，“ナビ” が，直交クラスタとして “GPS”，“WiFi”，“G0”，“小容量”，“大容量” がそれぞれ識別される．

4.3 非選択が他に影響を及ぼさないフィーチャ群の識別

4.3.1 付加サブ木

非選択が他に影響を及ぼさないフィーチャ群とは，それらを非選択としても，他のフィーチャの選択・非選択に影響を及ぼさないフィーチャ群である．本手法では，任意フィーチャを頂点とするサブ木のみを対象に，その非選択が他のフィーチャに影響を及ぼさないかどうかを判断する．影響を及ぼさない場合，それを付加サブ木と呼ぶ．

4.3.2 付加サブ木の識別

任意フィーチャを頂点とし，それ以下のどのフィーチャもフィーチャ選択に含まれておらず，また他の部分と CTC 関係で結ばれていなければ，その部分を付加サブ木と判断する．**図 6** にアルゴリズムを示す．

CTC の両端の親をたどる処理が含まれており，計算量は 4.2.3 項と同様に $O(N^3)$ である．

図 3 では “GPS” と “ナビ” が付加的なサブ木となる．

```
input: フィーチャモデル FM, フィーチャ選択 FC
       共通クラスタに含まれるフィーチャの集合: FCOM
       コアクラスタの頂点フィーチャの集合: FTOPcore
       直交クラスタの頂点フィーチャの集合: FTOPortho
output: フィーチャ構成 CONF
procedure:
1. コアクラスタからのフィーチャ構成導出
     FTOPortho 中のフィーチャを非選択として FC に加える;
     FM と FC からフィーチャ構成 CONF0 を導出する;
2. 直交クラスタからのフィーチャ構成導出
     FTOPcore 中のフィーチャを非選択としたフィーチャ
     選択を FCcore とする;
     FM と FCcore からフィーチャ構成 CONF1 を導出する;
3. 二つの和をとりフィーチャ構成 CONF を得る;
     CONF = CONF0 ∪ CONF1
```

図 7 部分毎のフィーチャ構成導出
Fig. 7 Derivation based on FM division.

4.4 フィーチャ構成の導出

4.4.1 部分ごとのフィーチャ構成導出

コアクラスタと直交クラスタそれぞれからフィーチャ構成を求め，それらの和集合を求めることで全体のフィーチャ構成を得る．**図 7** に手順を示す．この際，たとえば直交クラスタの頂点フィーチャを非選択とすることでコアクラスタのみからフィーチャ構成を得ることができる．逆も同様である．なおフィーチャ構成導出では，各クラスタに対して表 1 で示した対応関係で制約記述を作り制約ソルバで構成を導出する．

図 3 の場合，コアクラスタからはたとえば {携帯電話，通話，電池，メディア，カメラ，アプリ，地図} が，直交クラスタからはたとえば {携帯電話，通話，電池，大容量} がそれぞれ導出され，和をとることで {携帯電話，通話，電池，大容量，メディア，カメラ，アプリ，地図} が得られる．

フィーチャ構成導出を繰り返す際は，関心のあるクラスタのみ導出を繰り返し，他方のクラスタは 1 度求めたフィーチャ構成をそのまま使うことで，効率的な繰返しができる．この際，1 度生成されたフィーチャ構成を否定して制約に加えることで異なるフィーチャ構成を導出できる．

4.4.2 小さなサイズのフィーチャ構成導出

FM から付加サブ木を除外してフィーチャ構成を導出することで，小さなフィーチャ構成を得ることができる．**図 8** に手順を示す．

付加サブ木に含まれるフィーチャは “GPS” と “ナビ” であるから，これらは非選択となる．表 2 では#1，#3，#4 が相当するので，導出されるフィーチャ構成のサイズは 8 もしくは 9 となる．サイズ 10，11 のフィーチャ構成は導出されないため，通常の導出方法より小さなサイズのフィーチャ構成が得られる．ここで小さなフィーチャ構成とは余

input: フィーチャモデル *FM*, フィーチャ選択 *FC*
付加サブ木の頂点フィーチャの集合: *FOSUB*
output: フィーチャ構成 *CONF*
procedure:
FOSUB 中のフィーチャを非選択として *FC* に加える ;
FM と *FC* からフィーチャ構成 *CONF* を導出する

図 8 小さなフィーチャ構成の導出
Fig. 8 Derivation of smaller feature configuration.

分な付加サブ木を選択しないという意味であり，最小サイズ（この場合は 8）のフィーチャ構成の導出を保証するものではない．

なお 4.4.1 項の部分ごとのフィーチャ構成導出においても，付加サブ木を非選択とすることによって，クラスタごとに小さなサイズのフィーチャ構成を得ることができる．

5. 評価

本章では，提案手法の評価について述べる．

5.1 目的

提案手法に関して，以下を目的として，評価を行った．

- 本手法で識別できる直交クラスタや付加サブ木が，FM 中にどの程度存在するのかを確認する．直交クラスタや付加サブ木が存在しなければフィーチャ構成導出に活用することができない．様々な FM に対して本手法を適用して直交クラスタや付加サブ木の識別を試み，識別された直交クラスタや付加サブ木のサイズを調べる．
- 本手法で，フィーチャ構成導出が効果的に行われるかどうかを確認する．識別された直交クラスタや付加サブ木を利用することで，導出の繰返しや，小さなサイズのフィーチャ構成導出にどの程度効果があるのかを確認する．

5.2 直交クラスタと付加サブ木の割合

5.2.1 評価方法

評価はシミュレーションによって行った．本提案手法に基づきフィーチャ構造解析およびフィーチャ構成導出を行うプログラムを作成し，提案手法を用いて直交クラスタや付加サブ木の識別を行い，直交クラスタや付加サブ木のサイズ（フィーチャ数）を調べ，FM サイズに対する比率を求めた．

評価には FM 生成ツール BeTTy [22] によりランダムに作られた FM と，FM レポジトリ SPLOT [17], [23] に登録されている意味を持った FM との 2 種類を利用した．

- BeTTy は指定されたパラメータ（下記の NF や Rctc など）に基づいて，ランダムに FM を生成するものである．BeTTy を用いた評価では，いくつかのパラメータを変動させて，結果がどのように変化するのかという FM の構造と結果との関係性を見ることを目的とした．5.2.2 項 (1), (2) がこれに相当する．生成にあたっては，以下のパラメータを指定した．
  - NF：FM サイズ
  - Rctc：FM 中の全フィーチャ数に対する CTC の数の割合（たとえば FM サイズ 10 で，Rctc が 10%なら，1 つの CTC が生成される）
- SPLOT は，実際になんらかの対象を記述した FM のリポジトリのため，BeTTy の FM に比べてより現実的な FM である．一方パラメータ的にみると BeTTy のように様々なパターンが含まれているわけではない．SPLOT の FM を用いた評価では，現実的な FM に対してどういう結果が得られるかを確認することを目的とした．5.2.2 項 (3) がこれに相当する．

同じサイズの FM でも深さが深いものや幅が広いものなど多様である．特に本手法は，クラスタ単位でコア部分や直交部分の識別を行っているが，枝分かれの数でクラスタの数が変わり，結果に影響を及ぼすと考えられる．そこでルート直下の枝（子フィーチャ）の数（以下 NB と呼ぶ）による影響も調べた．

フィーチャ選択は，FM サイズ（FM 中の共通部分を含む全フィーチャ数）に対する選択するフィーチャ数の割合（以下，フィーチャ選択比率 Rfc と呼ぶ）を指定し，ランダムに生成した．なお Rfc は明示的に選択したフィーチャ数の比率であり，それを指定したことによって間接的に選ばれるフィーチャの数は含まない．

5.2.2 結果

条件を変えて直交クラスタや付加サブ木の比率を調べた．

(1) BeTTy：NF や NB との関係

BeTTy で FM サイズの違う FM を生成した（NF：10〜100）．1 つの NF に対して 100 個の FM を生成した．なお Rctc は 10%，Rfc は 10%とした．Rfc が 10%なので FM サイズが小さい場合は指定されるフィーチャ数も少ないが，3.2 節で記述したように多段的なフィーチャ構成導出を行う場合には，各段階では少数のフィーチャが指定されることもあるためこの設定で評価をした．なお，より大きな Rfc に関しては (2) で評価した．

各 FM に対して 10 回フィーチャ選択を生成して直交クラスタを識別し，そのサイズと FM サイズとの比率を求めた．**図 9** にその結果を箱ひげ図で示す．また下部に比率の平均値を示す．NF にかかわらず直交クラスタが存在しない FM が一定数存在している．

本手法はクラスタ単位で識別を行っているため，直感的には枝分かれが多く横長の（あるいは木の深さが浅い）FM の方が，クラスタが多くなり識別の可能性が高まると考えられる．図 9 のデータを，ルート直下の枝の数（NB）で分類して再集計したものが**図 10** である．NB が小さい場

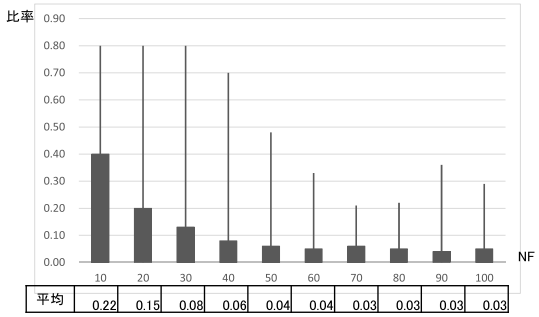

図 9 直交クラスタの比率（BeTTy，NF10-100）
Fig. 9 Orthogonal cluster ratio (BeTTy, NF10-100).

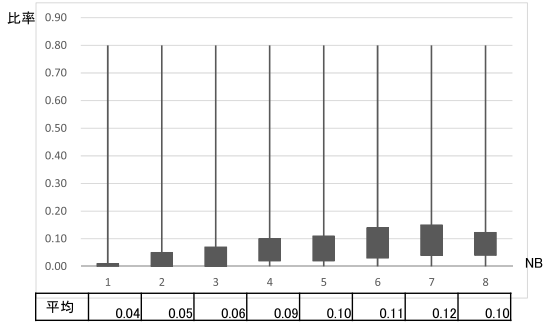

図 10 直交クラスタの NB ごとの比率（BeTTy，NF10-100）
Fig. 10 Orthogonal cluster ratio by NB (BeTTy, NF10-100).

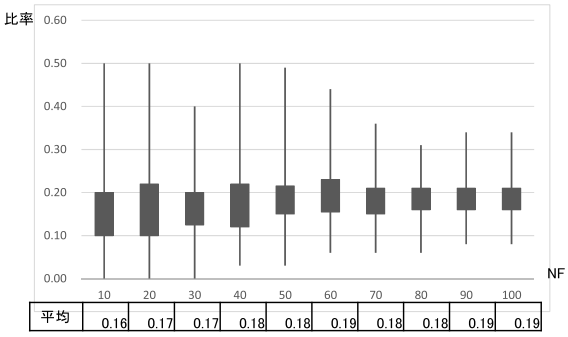

図 11 付加サブ木の比率（BeTTy，NF = 10-100）
Fig. 11 Optional sub-tree ratio (BeTTy, NF10-100).

合には直交クラスタが存在しない FM が多いが，NB が 4 以上では 75%以上の FM で存在しており，これらにおいては FM サイズに対する比率の平均は 10%程度である．

図 11 は上記と同じ条件で識別した付加サブ木の比率であり，**図 12** はそれを NB 単位に集計したものである．付加サブ木は NB にかかわらずほとんどの FM に存在し，平均値は 20%弱であることが確認された．

(2) BeTTy：Rctc や Rfc との関係

CTC に参加するフィーチャ数が増えると制約がきつくなり直交クラスタや付加サブ木の存在に影響すると考えられる．

図 13 に，(1) と同じ条件で，Rctc を 10%から 60%まで変化させた場合の，直交クラスタならびに付加サブ木の比率を示す．Rctc が高まるにつれ，存在する FM が減り，サイズの比率も減少している．

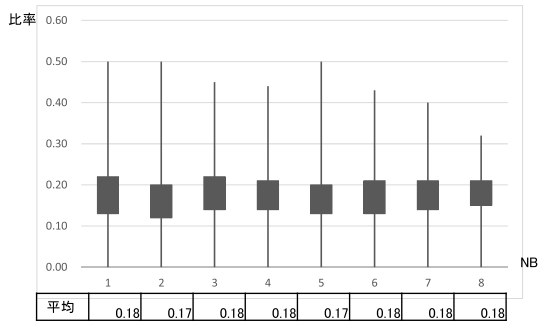

図 12 付加サブ木の NB ごとの比率（BeTTy，NF10-100）
Fig. 12 Optional subtree ratio by NB (BeTTy, NF10-100).

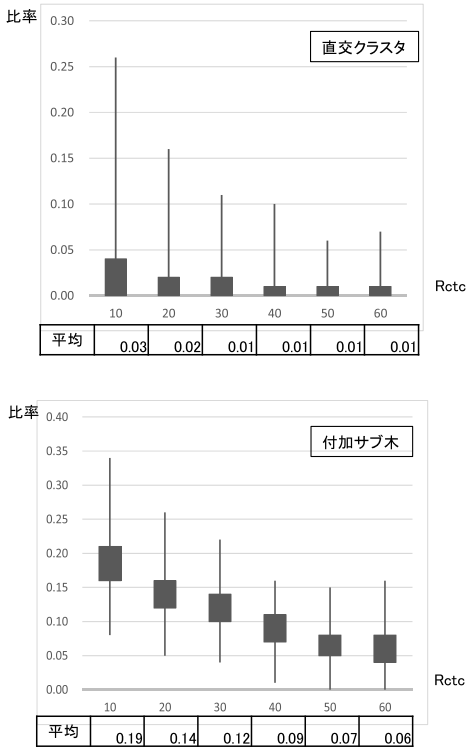

図 13 CTC の影響（BeTTy，NF = 100，Rctc10-60）
Fig. 13 Effect of CTC (BeTTy, NF=100, Rctc10-60).

フィーチャ選択に含まれるフィーチャ数も同様に影響を持つと考えられる．**図 14** に，(1) と同じ条件で，Rfc を 10%から 60%まで変化させた場合の，直交クラスタならびに付加サブ木の比率を示す．CTC と類似の傾向が確認できる．

(3) SPLOT リポジトリ

実験時点で SPLOT に登録されていた FM サイズ 70 以上の FM すべてと，NF = 10〜60 前後のものを均等に無作為に選択し，合計 80 の FM を評価に利用した．NF は 10 から 366 までである．これらを対象に Rfc = 10% で実験を行った．

直交クラスタの比率，付加サブ木の比率をそれぞれ**図 15** と**図 16** に示す．いずれも NB で集計した．

NB = 1, 4 以外では 75%以上の FM で直交クラスタが存

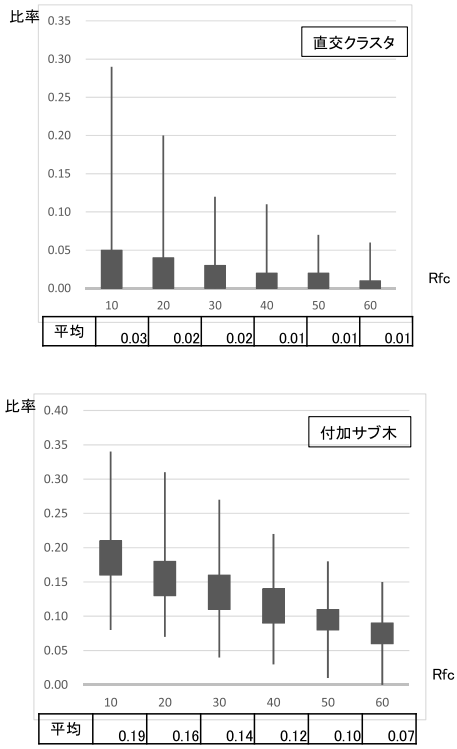

図 14 フィーチャ選択の影響（BeTTy，NF = 100，Rfc = 10-60）
Fig. 14 Effect of feature choice (BeTTy, NF = 100, Rfc = 10-60).

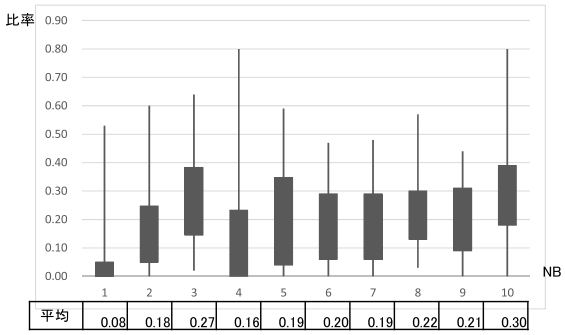

図 15 NB で集計した直交クラスタの比率（SPLOT）
Fig. 15 Orthogonal cluster ratio by NB (SPLOT).

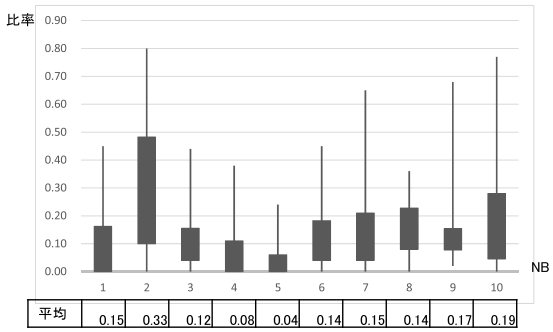

図 16 NB で集計した付加サブ木の比率（SPLOT）
Fig. 16 Optional subtree ratio by NB (SPLOT).

表 3 製品のリストアップ（上：BeTTy，下：SPLOT）
Table 3 Configuration list up (Upper: BeTTy, Lower: SPLOT).

従来-提案	NF				
	10	20	30	40	50
OK-OK	1.00	1.00	0.76	0.05	0
NG-OK	0	0	0.24	0.7	0.47
NG-NG	0	0	0	0.25	0.53

従来-提案	NF						
	10	20	30	40	50	60	70-366
OK-OK	1.00	0.93	0.72	0.34	0.29	0	0.12
NG-OK	0	0.07	0.22	0.61	0.44	0.72	0.34
NG-NG	0	0	0.06	0.04	0.27	0.28	0.54

在し，NB = 1, 4, 5 以外では付加サブ木が存在した．また直交クラスタの FM に対する平均比率は BeTTy の場合よりも大きく 20%程度だった．SPLOT は意味を持った FM であり，意味単位ごとにグループ化して構築されているからと推察される．

5.3 部分ごとのフィーチャ構成導出

部分ごとにフィーチャ構成導出をすることがフィーチャ構成の利用に実用上有効なことがあり，またそうした利用をする場合には，全体のリストアップをする場合に比べてリストアップできるケースが増えることが期待できることを指摘した（3.1 節）．ここでは，手法を実際に FM に適用することで，どの程度リストアップできるケースが増えるのかを評価した．1 つの FM に対して従来手法（FM 全体を 2.2 節で説明した方法で制約記述に変換してフィーチャ構成を求める方法）と，4.4.1 項で示した提案手法を用い，すべてのフィーチャ構成を導出した．ただし導出数の上限を 1,000 とし，それまでに数え上げが終わるかどうかを調べた（提案手法では 2 つの部分からの導出数の和の上限が 1,000）．

表 3 に結果を示す．OK-OK はどちらの手法でもすべて数え上げできたことを，NG-OK は従来手法では数え上げができなかったが提案手法ではできたことを，NG-NG はどちらの手法でも数え上げができなかったことを示す．各セルは，そのサイズの FM 中で該当する FM がどれだけあったかの比率を示す．表の数字の意味を正確に説明する．フィーチャ数 NF が n（$n = 10, 20, 30, \cdots$）の FM の集合を FM(n)，FM(n) に含まれる FM の中で，OK-OK であった FM の集合を $FM_{OK\text{-}OK}(n)$，NG-OK であった FM の集合を $FM_{NG\text{-}OK}(n)$，NG-NG であった FM の集合を $FM_{NG\text{-}NG}(n)$ とする．この場合，NF = n の OK-OK，NG-OK，NG-NG の各セルには $\#FM_{OK\text{-}OK}(n)/\#FM(n)$，$\#FM_{NG\text{-}OK}(n)/\#FM(n)$，$\#FM_{NG\text{-}NG}(n)/\#FM(n)$ が示さ

れている．なお各 FM の集合の要素数を集合名に#をつけて表現している．たとえば BeTTy の NF = 30 では，提案手法ではすべて FM の数え上げができたが，従来手法では 24%の FM で数え上げができなかったことを示す．このように BeTTy では NF が 30 以上，SPLOT では 20 以上で提案する部分ごとのフィーチャ構成導出でリストアップできるケースが増えていることが確認され，BeTTy では NF = 50，SPLOT では NF が 70-366 でも OK-OK と NG-OK をあわせて半分弱の FM がリストアップできるケースとなっている．

5.4 小さなサイズのフィーチャ構成導出

提案手法（4.4.2 項）を用いることで，どの程度小さなサイズのフィーチャ構成が導出できるのかを評価した．5.3 節同様に 1 つの FM に対して，従来手法と提案手法を用い，導出できるフィーチャ構成をすべて導出し，サイズの平均値の比率（従来手法に対する提案手法のサイズの縮小率）を調べた．

図 17 はサイズの縮小率を NF ごとに示したものである．上は BeTTy の FM，下は SPLOT の FM である．いずれの場合も，小さなサイズのフィーチャ構成を導出する効果が確認でき，平均の縮小率は 10%から 40%であった．

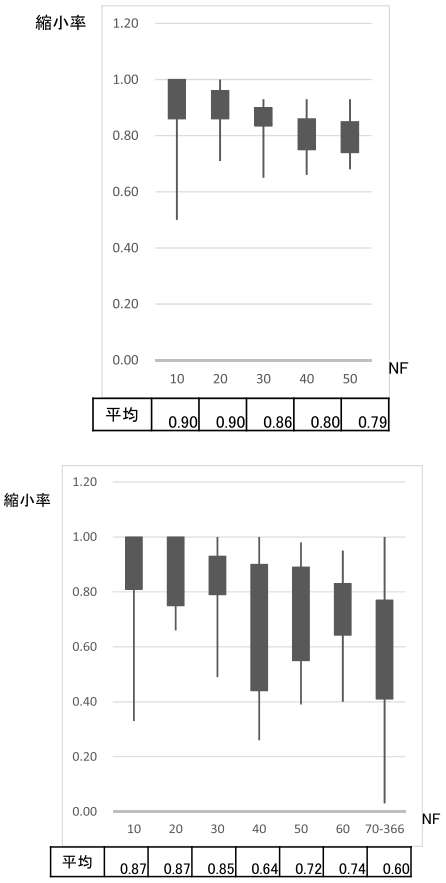

図 17 構成サイズの縮小率（上：BeTTy，下：SPLOT）
Fig. 17 Size reduction ratio (Upper: BeTTy, Lower: SPLOT).

なお BeTTy では，NF が 60 以上の FM の多くですべてのフィーチャ構成の導出が終わらなかったため，データを示していない．SPLOT では導出可能な FM を揃えることができ，すべてを示した．

5.5 結論

以上より，以下が確認できた．

- 多くの FM で直交クラスタが存在し，その全体への平均比率は BeTTy で 10%程度，SPLOT では 20%程度であった．付加サブ木についても多くの FM で存在し，BeTTy で 20%程度，SPLOT では 10〜20%程度であった．
- 直交クラスタの識別はクラスタ単位で行っているため，クラスタが少ないと存在しない場合が多くなる．CTC に参加するフィーチャや，フィーチャ選択に含まれるフィーチャの数の比率が増加しても制約がきつくなり存在しない場合が多くなる．
- 部分ごとのフィーチャ構成を導出することによって，リストアップできるケースが増えることが確認された．たとえば SPLOT で NF = 70 以上の場合，分割しないと 12%しかリストアップできないが，分割することで半分弱がリストアップできた．また小さなフィーチャ構成の導出によって，小さなフィーチャ構成が導出されることが確認できた．

6. 議論

6.1 近似的な解析

本研究では共通部分，コア部分，直交部分を調べるために近似的に共通クラスタ，コアクラスタ，直交クラスタを識別している．コアクラスタはクラスタ単位で判断されるため広く識別される．つまりコアクラスタ中のフィーチャ群の中には実際には直交部分あるいは共通部分に含まれるものが存在しうる．逆に共通クラスタと直交クラスタは狭く識別される．これにより，フィーチャ構成の導出を正しく行うことができる．

6.2 提案手法の有効性

提案手法は FM の構造上の特徴に基づいて解析を行うため，FM の構造によってその効果が変わる．特にクラスタが少ない場合には識別できる部分が小さくなる．また CTC によって多くのクラスタが接続されたり，多くのクラスタ中のフィーチャがフィーチャ選択に含まれたりしても識別できる部分が小さくなる．しかしながら評価実験からは，多くの場合で直交クラスタや付加サブ木が存在し，導出を効果的に行えることが確認された．

6.3 より厳密な解析

本手法では FM を共通部分直下のサブ木に分割し，フィー

チャ選択に関わるサブ木と CTC で接続されていないサブ木を直交部分として識別している．こうすることでコア部分と直交部分を容易に分離できるからである．しかしながら CTC で接続されていても直交する状況もありうる．こうした CTC で接続されている部分を，導出される構成に影響しないように正しく分離するためには，本手法のように FM のモデル要素の接続関係からだけで判断するのではなく，命題論理に立ち入ったより厳密な解析が必要となる．たとえば FM のスライシングを用いることで，CTC で接続されていても 2 つの部分に正しく分離することが可能となる．それによってコア部分をより小さくできるならば提案手法を改善する可能性がある．しかしながら，提案手法では発見できない直交部分を発見し，コア部分をより小さくする具体的な方法は知られていない．

また厳密な解析はそれだけ計算量が高くなり，たとえば FM のスライシングは NP 困難である．もちろん計算量が高いからといってつねに時間がかかるわけではなく，現実的な時間で処理できる場合もある．しかしながら提案手法は簡易な識別を行っているため安定して高速に処理が可能であり，かつ一定の有効性が確認されるため，1 つの現実解と考える．

このように，命題論理に立ち入った厳密な方法を利用して提案手法を改善する可能性はあると考えられるが，その具体的な方法は今後の検討課題である．

6.4 妥当性への脅威

評価実験では BeTTy と SPLOT の FM を利用した．BeTTy で生成される FM はランダムに生成されるため場合によっては奇異な FM が生成されることもある．一方 SPLOT は意味のある FM であるが，例題的な FM も多く含まれている．さらに異なったタイプの FM での評価も望まれる．

1 つの FM に対して可能なフィーチャ選択の数は膨大にある．今回はフィーチャ選択をランダムに生成したが，その妥当性も要検討である．実際に行われるフィーチャ選択の特徴などを検討することで，フィーチャ選択の方法を改善することも考えられる．

フィーチャ構成導出の評価では，多くのあるいはすべてのフィーチャ構成を導出するなどしたため，評価に多くの時間がかかった．SPLOT では FM サイズ 300 程度まで評価をしたが，評価方法を改善してより大きなサイズの FM に対して評価することも必要と考える．

7. 関連研究

FM は SPL 開発などで広く使われており，フィーチャ構成導出は，製品導出 [9] などの際に必要となる重要な作業である．フィーチャ構成導出には SAT や CSP による方法が提案されている [4], [17], [18], [25]．また形式性に基づく FM の解析に関する研究もある [3]．しかしながら FM を扱うための計算量は一般に高く [21]，スケーラブルな導出方法 [20] や，高速なフィーチャ構成のリストアップ方法 [10] などの提案がある．そうした中，本稿では FM 中のコア部分や直交部分，あるいは選択しなくてもよいフィーチャ群を識別することで，フィーチャ構成導出に活用する手法を提案した．

FM を部分に分割するという観点からは，FM のモジュール化に関する研究がある [2], [12]．FM が独立して扱えるモジュールに分割できるなら様々な処理が容易になることが期待される．今回の評価で SPLOT の方が直交クラスタのサイズが多かったことは，意味的なグループ化などが行われているからだと推察される．有用なモジュール化に関しては，さらに検討が必要と考える．

本研究は，FM 定義時点でモジュール化を行うのではなく，導出時に必要な部分を識別するものである．こうした識別に関しては FM のスライシング [1], [15] の研究があるが，その計算量は高いことが知られている．本手法は厳密にスライスを求めるのではなく，低計算量で近似的に直交部分などを求める手法を提案し，多くの状況でフィーチャ構成導出に有用に利用できることを示した．Garlan はソフトウェア工学の一般的な方向性として，不確かさのある状況では過度の厳密さよりそれなりに有用な近似解を求めることの重要性を指摘したが [11]，本研究はフィーチャ構成導出において，類似した立場をとるものである．

段階的導出 [8] はフィーチャ選択を多段的に行いながら最終的な構成を求める手法である．本手法を応用し直交クラスタや付加サブ木を “不明” とすることで，決定部分を少なくして次の段階での選択肢を多くするという応用が可能である．なお Chen らは，できるだけ大きな決定を行うことで導出ステップを減らすという逆の提案をしているが [5]，これはフィーチャに対する選択・非選択の決定を 1 つずつ行う状況を想定しており，本稿の想定とは異なる．

8. おわりに

本稿では，FM の近似的な解析に基づきフィーチャ構成を導出する手法について提案し，シミュレーションによってその効果を確認した．実際のフィーチャ構成導出，あるいはそれを利用した製品導出などへの適用や評価は今後の課題である．

参考文献

[1] Acher, M., Collet, P., Lahire, P. and France, R.B.: Slicing Feature Models, *Proc. ASE 2011*, pp.424–427 (2011).
[2] Bagheri, E., Ensan, F., Gasevic, D. and Boskovic, M.: Modular feature models: Representation and configuration [online], *Journal of Research and Practice in Information Technology*, Vol.43, No.2, pp.109–140 (2011).
[3] Benavides, D., Trinidad, P. and Ruiz-Cortés, A.: Au-

tomated reasoning on feature models, *Proc. CAiSE05*, LNCS, Vol.3520, pp.491–503 (2005).
[4] Batory, D.: Feature Models, Grammars, and Propositional Formulas, *Proc. SPLC 2005*, pp.7–20 (2005).
[5] Chen, S. and Erwig, M.: Optimizing the Product Derivation Process, *Proc. SPCL 2011*, pp.35–44 (2011).
[6] Clements, P. and Northrop, L.: *Software Product Lines: Practices and Patterns*, Addison-Wesley (2001).
[7] Czarnecki, K., Helsen, S. and Eisenecker, U.W.: Formalizing cardinality-based feature models and their specialization, *Software Process: Improvement and Practice*, Vol.10, No.1, pp.7–29 (2005).
[8] Czarnecki, K., Helsen, S. and Eisenecker, U.W.: Staged configuration through specialization and multilevel configuration of feature models, *Software Process: Improvement and Practice*, Vol.10, No.2, pp.143–169 (2005).
[9] Deelstra, S., Sinnema, M. and Bosch, J.: Product derivation in software product families: A case study, *Journal of Systems and Software*, Vol.74, No.2, pp.173–194 (2005).
[10] Galindo, J.A., Acher, M., Tirado, J.M., Vidal, C., Baudry, B. and Benavides, D.: Exploiting the Enumeration of All Feature Model configurations, *Proc. SPLC 2016*, pp.74–48 (2016).
[11] Garlan, D.: Software Engineering in an Uncertain World, *FoSER'10*, pp.125–128 (2010).
[12] Kaestner, C., Apel, S. and Ostermann, K.: The Road to Feature Modularity?, *Proc. SPLC 2011*, Vol.2 (2011).
[13] Kang, K., Cohen, S., Hess, J., Novak, A.W. and Peterson, S.: Feature-oriented domain analysis (FODA) feasibility study, *CMU/SEI-90-TR-21* (1990).
[14] 岸　知二，野田夏子：近似化によるフィーチャモデルからの製品導出法，*KBSE 2016*, No.KBSE2016-41, pp.13–18 (2016).
[15] Krieter, S., Schroeter, R., Thuem, T., Fenske, W. and Saake, G.: Comparing Algorithms for Efficient Feature-Model Slicing, *Proc. SPLC 2016*, pp.60–64 (2016).
[16] Le, D. M., Lee, H., Kang, K.C. and Keun, L.: Validating Consistency between Feature Model and Its Implementation, *ICSR 2013*, pp.1–16 (2013).
[17] Mendonca, M., Branco, M. and Cowan, D.: S.P.L.O.T.—Software Product Lines Online Tools, *Companion to the 24th ACM SIGPLAN International Conference on OOPSLA 2009* (2009).
[18] Mendonca, M., Wasowski, A. and Czarnecki, K.: SAT-based analysis of feature models is easy, *Proc. SPLC 2009*, pp.231–240 (2009).
[19] Santos, A.R. and de Almeida, E.S.: Do #ifdef-based Variation Points Realize Feature Model Constraints?, *ACM SIGSOFT, Software Engineering Notes*, Vol.40, No.6, pp.1–5 (2015).
[20] Sayyad, S., Ingram, J., Menzies, T. and Ammar, H.: Scalable Product Line Configuration: A Straw to Break the Camel's Back, *Proc. ASE 2013*, pp.465–474 (2013).
[21] Schobbens, P.Y., Heymans, P. and Trigaux, J.C.: Feature diagrams: A survey and a formal semantics, *Proc. RE2006*, pp.139–148 (2006).
[22] Segura, S., Galindo, J.A., Benavides, D., Parejo, J.A. and Ruiz-Cortés A.: BeTTy: Benchmarking and Testing on the Automated Analysis of Feature Models, *Proc. VaMoS'12*, pp.63–71 (2012).
[23] available from ⟨http://ec2-52-32-1-180.us-west-2.compute.amazonaws.com:8080/SPLOT/index.html⟩.
[24] Tamura, M., Taga, A., Kitagawa, S. and Banbara, M.: Compiling finite linear CSP into SAT, *Constraints*, Vol.14, No.2, pp.254–272 (2009).
[25] White, J., Schmidt, D.C., Benavides, D., Trinidad, P. and Ruiz-Cortés, A.: Automated Diagnosis of Product-line Configuration Errors in Feature Models, *Proc. SPLC2008*, pp.225–234 (2008).

岸 知二 （正会員）

京都大学大学院工学研究科情報工学専攻修了．北陸先端科学技術大学院大学博士後期課程修了．博士（情報科学）．NEC，北陸先端科学技術大学院大学を経て，2009 年より早稲田大学教授．

野田 夏子 （正会員）

東京女子大学大学院理学研究科数学専攻修了．北陸先端科学技術大学院大学博士後期課程修了．博士（情報科学）．NEC 勤務を経て，2013 年より芝浦工業大学准教授．